Matematica delle Reti Neurali

Volume VI

Silvano Salvador

Modelli di Stato · Apprendimento per Rinforzo · Teoria PAC
Mixture of Experts · Kernel Avanzati · Algoritmi Evolutivi · Teoria della Misura

Sezioni §342–380 — Fondamenti avanzati dell'apprendimento automatico moderno

HMMFiltro di KalmanParticle Filter MDP & BellmanPPORademacher PAC-BayesMoECMA-ES NOTEARSMartingaliDeep Kernel Learning

Modelli di Markov Nascosti: Struttura, Parametri e Tre Problemi Fondamentali

La formalizzazione probabilistica delle sequenze osservabili generate da stati latenti discreti — dalla riconoscimento vocale alla bioinformatica.

342.1 · La Struttura degli HMM e la Proprietà di Markov

Un **Modello di Markov Nascosto (HMM)** è il modello probabilistico generativo più importante per le sequenze discrete, e al tempo stesso uno degli esempi più illuminanti di come una struttura grafica semplice possa condensare un'enorme quantità di complessità computazionale in algoritmi efficienti. L'idea fondamentale è la separazione tra ciò che si osserva e ciò che governa la generazione: esiste una sequenza di *stati latenti* $z_1, z_2, \ldots, z_T$ che evolve markovianamente — ogni stato dipende solo dal precedente — e una sequenza di *osservazioni* $x_1, x_2, \ldots, x_T$ che dipendono ciascuna solo dallo stato corrente. Formalmente, un HMM è specificato da: lo spazio degli stati latenti $\mathcal{Z} = \{1, \ldots, K\}$ con K stati discreti; la distribuzione iniziale $\pi_k = P(z_1 = k)$; la matrice di transizione $A_{ij} = P(z_t = j \mid z_{t-1} = i)$ con righe che sommano a uno; e la matrice di emissione $B_{kl} = P(x_t = l \mid z_t = k)$ (per osservazioni discrete) oppure una famiglia di distribuzioni $p(x_t \mid z_t = k)$ (per osservazioni continue, tipicamente gaussiane miste). Il modello congiunto si fattorizza grazie alla proprietà di Markov e all'indipendenza condizionale delle osservazioni dagli stati:

$$P(x_1, \ldots, x_T, z_1, \ldots, z_T) = \pi_{z_1} B_{z_1 x_1} \prod_{t=2}^{T} A_{z_{t-1} z_t} B_{z_t x_t}$$

Tre problemi distinti si pongono naturalmente sugli HMM, e ciascuno richiede un algoritmo dedicato. Il **problema della valutazione** chiede: data la sequenza di osservazioni $\mathbf{x} = (x_1, \ldots, x_T)$ e i parametri del modello, qual è la probabilità della sequenza $P(\mathbf{x})$? Il cal-

colo diretto sommerebbe su tutti i K^T possibili percorsi di stati latenti — proibitivo anche per $K = 10$ e $T = 100$ — ma l'algoritmo forward (§343) lo risolve in $O(K^2T)$. Il **problema del decodaggio** chiede: qual è la sequenza di stati latenti più probabile data la sequenza di osservazioni? La risposta è l'algoritmo di Viterbi, che calcola il percorso MAP in $O(K^2T)$. Il **problema dell'apprendimento** chiede: dati i dati di training, come stimare i parametri (π, A, B)? La soluzione è l'algoritmo di Baum-Welch (§343), un'istanza dell'algoritmo EM per i modelli grafici.

342.2 · Applicazioni Storiche e Sviluppi Moderni

Gli HMM furono introdotti da Baum e Petrie (1966) nel contesto della matematica statistica, ma divennero praticamente rilevanti con il lavoro di Jelinek, Bahl e Mercer alla IBM negli anni Settanta, dove costituirono la spina dorsale dei primi sistemi di riconoscimento vocale a grande vocabolario. Per vent'anni — dalla metà degli anni Settanta alla fine degli anni Novanta — tutti i sistemi competitivi di riconoscimento del parlato erano basati su HMM con emissioni gaussiane miste (GMM-HMM), e il codice sviluppato durante quel periodo è ancora parzialmente in uso nei sistemi ibridi GMM-HMM/DNN. In bioinformatica, gli HMM sono alla base degli *Hidden Markov Model Profile* (HMM-Profile, Krogh et al. 1994) per il confronto tra sequenze proteiche — lo strumento HMMER è tutt'oggi lo standard per la ricerca di omologhi nelle banche dati genomiche. I **Profile HMM** modellano una famiglia di proteine allineate come un HMM dove ogni colonna dell'allineamento corrisponde a uno stato di emissione, con stati speciali per le inserzioni e le delezioni — una struttura che cattura la variabilità evolutiva di una famiglia proteica in modo molto più espressivo di un semplice profilo di frequenza amminoacidica. La transizione verso i modelli neurali non ha eliminato gli HMM ma li ha integrati: nei sistemi DNN-HMM per il riconoscimento vocale, la rete neurale produce le probabilità di emissione condizionate al contesto acustico, mentre la struttura temporale è ancora governata dalla matrice di transizione dell'HMM.

Algoritmo Forward-Backward e Baum-Welch

Calcolo efficiente delle marginali posteriori nelle sequenze — il messaggio forward, il messaggio backward e l'algoritmo EM per l'apprendimento.

343.1 · L'Algoritmo Forward: Probabilità Prefissa in Tempo Lineare

Il calcolo di $P(\mathbf{x})$ tramite somma diretta su tutti i percorsi di stati è intrattabile, ma la struttura a catena del modello di Markov permette una dinamica programmazione che riduce il costo a $O(K^2T)$. Si definisce la **variabile forward** $\alpha_t(k) = P(x_1, \ldots, x_t, z_t = k)$ — la probabilità congiunta di aver generato le prime t osservazioni e di trovarsi nello stato k al tempo t. La ricorrenza è intuitiva: la probabilità di essere nello stato k al tempo t avendo generato $x_1, \ldots, x_t$ si ottiene sommando, su tutti gli stati j al tempo $t-1$, la probabilità di essere stati in j, di essere transizionati verso k, e di aver emesso x_t:

$$\alpha_t(k) = B_{k,x_t} \sum_{j=1}^{K} \alpha_{t-1}(j)\, A_{jk}, \qquad \alpha_1(k) = \pi_k B_{k,x_1}$$

La probabilità della sequenza completa è $P(\mathbf{x}) = \sum_k \alpha_T(k)$. In pratica si lavora sempre in scala logaritmica o con normalizzazione progressiva per evitare il *vanishing* numerico: prodotti di T probabilità minori di uno tendono a zero a macchina per T anche solo dell'ordine di poche decine, rendendo impossibile il calcolo diretto senza precauzioni. La normalizzazione a ogni passo — dividere α_t per $\sum_k \alpha_t(k)$ e salvare i fattori di scala — permette di calcolare $\log P(\mathbf{x}) = \sum_t \log c_t$ come somma dei logaritmi dei fattori di scala, mantenendo la stabilità numerica per sequenze arbitrariamente lunghe.

343.2 · L'Algoritmo Backward e le Probabilità Posteriori

Simmetricamente, si definisce la **variabile backward** $\beta_t(k) = P(x_{t+1}, \ldots, x_T \mid z_t = k)$ — la probabilità delle osservazioni future dato lo stato corrente. La ricorrenza percorre la sequenza all'indietro:

$$\beta_t(k) = \sum_{j=1}^{K} A_{kj}\, B_{j,x_{t+1}}\, \beta_{t+1}(j), \qquad \beta_T(k) = 1 \;\forall k$$

Con le variabili forward e backward si calcolano le **probabilità posteriori** degli stati e delle transizioni — le quantità necessarie per l'algoritmo di Baum-Welch. La probabilità marginale posteriore di essere nello stato k al tempo t è semplicemente proporzionale al prodotto: $\gamma_t(k) = P(z_t = k \mid \mathbf{x}) \propto \alpha_t(k)\,\beta_t(k)$. La probabilità congiunta posteriore di transizionare da j a k tra i tempi t e $t+1$ è $\xi_t(j,k) \propto \alpha_t(j)\, A_{jk}\, B_{k,x_{t+1}}\, \beta_{t+1}(k)$. Queste due quantità costituiscono il *passo E* dell'algoritmo di **Baum-Welch**, che è esattamente l'EM per gli HMM: il passo M

aggiorna i parametri come medie pesate dalle statistiche posteriori: $\hat{A}_{jk} = \sum_t \xi_t(j,k) / \sum_t \gamma_t(j)$ e $\hat{B}_{kl} = \sum_t \gamma_t(k)\,\mathbf{1}[x_t = l] / \sum_t \gamma_t(k)$. La convergenza al massimo locale della log-likelihood è garantita dalla teoria dell'EM.

343.3 · L'Algoritmo di Viterbi come Max-Product

L'algoritmo di **Viterbi** (1967) calcola la sequenza di stati più probabile $\hat{\mathbf{z}} = \arg\max_{z_1,\dots,z_T} P(\mathbf{z} \mid \mathbf{x})$ sostituendo le somme con i massimi nella ricorrenza forward. Si definisce $\delta_t(k) = \max_{z_1,\dots,z_{t-1}} P(x_1,\dots,x_t, z_t = k)$ e si aggiorna $\delta_t(k) = B_{k,x_t} \max_j [\delta_{t-1}(j)\, A_{jk}]$. Il backpointer $\psi_t(k) = \arg\max_j [\delta_{t-1}(j)\, A_{jk}]$ memorizza, per ogni stato e ogni tempo, da quale stato precedente si è arrivati percorrendo il percorso più probabile. La sequenza ottimale si recupera con un backtracking da $\hat{z}_T = \arg\max_k \delta_T(k)$ all'indietro. Come discusso nella §279, l'algoritmo di Viterbi è esattamente l'algoritmo Max-Product applicato al factor graph dell'HMM — una delle prime e più eleganti istanze della famiglia belief propagation che poi si è estesa ben oltre il contesto delle catene di Markov.

Il Filtro di Kalman: Inferenza Esatta nei Modelli Lineari-Gaussiani

La forma chiusa dell'aggiornamento bayesiano per sistemi dinamici lineari — predizione e correzione, guadagno di Kalman, applicazioni al tracking.

344.1 · Il Modello di Spazio degli Stati Lineare-Gaussiano

Il **filtro di Kalman** (Rudolf E. Kálmán, 1960) è forse il risultato più importante della teoria dell'ottima stima applicata ai sistemi dinamici, con applicazioni che spaziano dalla navigazione dei missili balistici (l'applicazione originale) alla guida autonoma, dalla fusione di dati sensoriali nel tracking di oggetti al livellamento delle curve finanziarie. Il filtro di Kalman è l'analogo continuo-gaussiano dell'HMM: il modello di spazio degli stati ha stati latenti *continui* $\mathbf{z}_t \in \mathbb{R}^d$ e osservazioni $\mathbf{x}_t \in \mathbb{R}^m$ che soddisfano equazioni lineari con rumore gaussiano:

$$\mathbf{z}_t = F\mathbf{z}_{t-1} + \mathbf{q}_t, \qquad \mathbf{q}_t \sim \mathcal{N}(\mathbf{0}, Q)$$

$$\mathbf{x}_t = H\mathbf{z}_t + \mathbf{r}_t, \qquad \mathbf{r}_t \sim \mathcal{N}(\mathbf{0}, R)$$

dove F è la matrice di transizione dello stato (dinamica del sistema), H è la matrice di osservazione (sensore), Q è la covarianza del rumore di processo e R quella del rumore di misura. La chiave della trattabilità analitica è la struttura lineare-gaussiana: la distribuzione posteriore di $\mathbf{z}_t$ dati tutti i dati fino a t rimane gaussiana per induzione, con parametri aggiornabili in forma chiusa. Questa è una conseguenza diretta del fatto che le trasformazioni lineari e le somme di gaussiane producono ancora gaussiane — una proprietà di chiusura che non vale per distribuzioni non gaussiane né per sistemi non lineari, come vedremo nella §345.

344.2 · Le Due Fasi del Filtro: Predizione e Correzione

Il filtro di Kalman alterna due fasi ad ogni passo temporale. La **fase di predizione** propaga la distribuzione posteriore dall'istante precedente all'istante corrente usando le equazioni del modello di transizione, prima di incorporare la nuova osservazione:

$$\hat{\mathbf{z}}_{t|t-1} = F\hat{\mathbf{z}}_{t-1|t-1}$$

$$P_{t|t-1} = FP_{t-1|t-1}F^T + Q$$

La **fase di correzione** incorpora la nuova osservazione $\mathbf{x}_t$ aggiornando la stima tramite il **guadagno di Kalman** K_t, che bilancia ottimalmente l'incertezza del predittore e quella della misura:

$$K_t = P_{t|t-1}H^T(HP_{t|t-1}H^T + R)^{-1}$$

$$\hat{\mathbf{z}}_{t|t} = \hat{\mathbf{z}}_{t|t-1} + K_t(\mathbf{x}_t - H\hat{\mathbf{z}}_{t|t-1})$$

$$P_{t|t} = (I - K_tH)P_{t|t-1}$$

La quantità $\boldsymbol{\nu}_t = \mathbf{x}_t - H\hat{\mathbf{z}}_{t|t-1}$ è chiamata *innovazione* o residuo di predizione: è la differenza tra l'osservazione effettiva e la sua previsione basata sulla stima corrente dello stato. Il guadagno di Kalman K_t dice quanto "fidarsi" dell'innovazione rispetto alla predizione: se R è grande (misure molto rumorose), K_t è piccolo e l'aggiornamento è conservativo; se $P_{t|t-1}$ è grande (molta incertezza sulla previsione), K_t è grande e ci si fida maggiormente della misura. Questa interpretazione bayesiana è precisa: il filtro di Kalman è l'aggiornamento bayesiano esatto per i modelli lineari-gaussiani, e la sua soluzione coincide con lo stimatore a varianza minima non distorto (MVUE) e con lo stimatore di massima verosimiglianza congiunto.

344.3 · Il Filtro di Kalman come Caso Speciale del Belief Propagation

Come discusso nella §279, il Sum-Product su factor graph ad albero calcola le marginali esatte. Il modello di spazio degli stati lineare-gaussiano è un factor graph ad albero (la catena temporale non ha cicli), e il Sum-Product su questo grafo produce esattamente il filtro di Kalman per il passo forward. Il passo backward corrisponde al **Kalman Smoother** (algoritmo RTS — Rauch, Tung e Striebel 1965): invece di calcolare $P(\mathbf{z}_t \mid x_1, \ldots, x_t)$ (filtro forward), calcola $P(\mathbf{z}_t \mid x_1, \ldots, x_T)$ usando l'intera sequenza di osservazioni — la distribuzione posteriore "lisciata" che è più accurata del filtro forward ma richiede di elaborare l'intera sequenza prima di produrre stime. L'applicazione più immediata del Kalman Smoother è l'algoritmo EM per i modelli di spazio degli stati: il passo E calcola le distribuzione posteriori lisciate $P(\mathbf{z}_t \mid \mathbf{x}_{1:T})$ e $P(\mathbf{z}_t, \mathbf{z}_{t+1} \mid \mathbf{x}_{1:T})$, e il passo M aggiorna le matrici F, H, Q, R come soluzioni di problemi di minimi quadrati ponderati.

Filtro di Kalman Esteso e Unscented

L'approssimazione del filtro di Kalman a sistemi non lineari — linearizzazione vs. propagazione sigma-points, l'errore di troncamento e i limiti dell'approssimazione gaussiana.

345.1 · La Rottura della Linearità e il Filtro Esteso

La maggior parte dei sistemi dinamici di interesse pratico — sistemi robotici, navigazione inerziale, dinamica atmosferica, modelli farmacocinetici — è non lineare: le equazioni di transizione e di osservazione hanno la forma $\mathbf{z}_t = f(\mathbf{z}_{t-1}) + \mathbf{q}_t$ e $\mathbf{x}_t = h(\mathbf{z}_t) + \mathbf{r}_t$ con f e h funzioni non lineari arbitrarie. Per questi sistemi, la distribuzione posteriore di $\mathbf{z}_t$ non è più gaussiana anche se la distribuzione iniziale lo è — le trasformazioni non lineari "deformano" la gaussiana in distribuzioni con code pesanti, multimodalità, e asimmetrie che l'approssimazione gaussiana

non può catturare. Il **Filtro di Kalman Esteso (EKF)** adotta la soluzione più semplice: linearizzare le funzioni f e h attorno alla stima corrente tramite espansione di Taylor al primo ordine. La matrice jacobiana $F_t = \partial f/\partial \mathbf{z}\big|_{\hat{\mathbf{z}}_{t-1|t-1}}$ sostituisce la matrice di transizione nel filtro lineare, e analogamente $H_t = \partial h/\partial \mathbf{z}\big|_{\hat{\mathbf{z}}_{t|t-1}}$ per la matrice di osservazione. L'EKF è computazionalmente efficiente — il costo è solo quello del calcolo dei jacobiani — ma introduce un **errore di linearizzazione** che può essere arbitrariamente grande per sistemi fortemente non lineari o quando la covarianza della distribuzione posteriore è ampia rispetto alla scala delle non-linearità. In pratica, l'EKF può essere instabile o divergente per sistemi con curvature elevate nelle funzioni di transizione.

345.2 · Il Filtro Unscented: Propagazione Deterministica tramite Sigma-Points

Il **Filtro di Kalman Unscented (UKF)** (Julier e Uhlmann 1997) risolve il problema dell'EKF senza calcolare jacobiani, tramite la tecnica della **Trasformazione Unscented**: invece di linearizzare la funzione non lineare, si rappresenta la distribuzione gaussiana corrente con un insieme di punti deterministici — i *sigma-points* — che catturano la media e la covarianza della distribuzione, e si propagano questi punti attraverso la funzione non lineare esatta, ricostruendo poi la media e la covarianza degli output. Per una distribuzione gaussiana $\mathcal{N}(\hat{\mathbf{z}}, P)$ in $\mathbb{R}^d$, si calcolano $2d+1$ sigma-points:

$$\boldsymbol{\sigma}^{(0)} = \hat{\mathbf{z}}, \quad \boldsymbol{\sigma}^{(i)} = \hat{\mathbf{z}} + (\sqrt{(d+\kappa)P})_i, \quad \boldsymbol{\sigma}^{(d+i)} = \hat{\mathbf{z}} - (\sqrt{(d+\kappa)P})_i$$

dove $(\sqrt{(d+\kappa)P})_i$ è l'i-esima colonna della radice quadrata matriciale. Si propagano poi i sigma-points attraverso f: $\boldsymbol{\sigma}_+^{(i)} = f(\boldsymbol{\sigma}^{(i)})$, e si ricostruisce la distribuzione degli output come combinazione pesata: $\hat{\mathbf{z}}_+ = \sum_i W_i^m \boldsymbol{\sigma}_+^{(i)}$ e $P_+ = \sum_i W_i^c (\boldsymbol{\sigma}_+^{(i)} - \hat{\mathbf{z}}_+)(\boldsymbol{\sigma}_+^{(i)} - \hat{\mathbf{z}}_+)^T + Q$. Il vantaggio cruciale dell'UKF è che cattura la media e la covarianza delle trasformazioni non lineari fino al terzo ordine in d — ben oltre il primo ordine dell'EKF — senza calcolare alcuna derivata analitica. L'UKF è per questo preferibile all'EKF ogni volta che il calcolo dei jacobiani è costoso (funzioni implicitamente definite, simulatori numerici) o quando l'accuratezza del terzo ordine è rilevante.

Filtri Particellari: Campionamento Sequenziale Monte Carlo

Filtraggio esatto per distribuzioni arbitrarie — campionamento per importanza sequenziale, la crisi dell'impoverimento e le tecniche di ricampionamento.

346.1 · Sequential Importance Sampling

Tanto l'EKF quanto l'UKF approssimano la distribuzione posteriore con una gaussiana — un'approssimazione inadeguata quando la vera distribuzione è multimodale, ha code pesanti, o ha una forma complessa. I **filtri particellari** (o Sequential Monte Carlo, SMC) abbandonano l'approssimazione gaussiana e rappresentano la distribuzione posteriore con un insieme di N particelle $\{\mathbf{z}_t^{(i)}, w_t^{(i)}\}_{i=1}^N$, dove ogni particella è un campione candidato dello stato e il suo peso $w_t^{(i)}$ riflette la sua probabilità relativa. L'aspettativa di qualunque funzione $f(\mathbf{z}_t)$ è stimata come media pesata: $\mathbb{E}[f(\mathbf{z}_t) \mid \mathbf{x}_{1:t}] \approx \sum_i w_t^{(i)} f(\mathbf{z}_t^{(i)})$. La tecnica di **Sequential Importance Sampling (SIS)** aggiorna ricorsivamente i pesi delle particelle ad ogni nuova osservazione: partendo da campioni della distribuzione di proposta $q(\mathbf{z}_t \mid \mathbf{z}_{t-1}^{(i)}, \mathbf{x}_t)$, i pesi vengono aggiornati in accordo con il rapporto di importanza $w_t^{(i)} \propto w_{t-1}^{(i)} \cdot p(\mathbf{x}_t \mid \mathbf{z}_t^{(i)}) \cdot p(\mathbf{z}_t^{(i)} \mid \mathbf{z}_{t-1}^{(i)}) / q(\mathbf{z}_t^{(i)} \mid \mathbf{z}_{t-1}^{(i)}, \mathbf{x}_t)$.

346.2 · Il Collasso dei Pesi e il Ricampionamento

Il problema principale del SIS puro è la **degenerazione dei pesi**: dopo pochi passi, quasi tutto il peso si concentra su una singola particella mentre tutte le altre hanno pesi trascurabili — il filtro diventa effettivamente monoparticellare pur mantenendo il costo computazionale dell'N-particellare. Questo fenomeno è inevitabile per $t \to \infty$ ed è quantificato dalla *effective sample size* (ESS): $\hat{N}_{\text{eff}} = 1/\sum_i (w_t^{(i)})^2$ cade drasticamente quando i pesi si concentrano. La soluzione classica è il **ricampionamento**: quando $\hat{N}_{\text{eff}}$ scende sotto una soglia (tipicamente $N/2$), si ricampionano N nuove particelle con rimpiazzo dalla distribuzione corrente, assegnando pesi uniformi $1/N$ alle nuove particelle. Il ricampionamento "elimina" le particelle a basso peso e duplica quelle ad alto peso, mantenendo la rappresentatività della distribuzione al costo di una riduzione della diversità delle particelle. La scelta della distribuzione di proposta ottimale $q^*(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{x}_t) = p(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{x}_t)$ (la posterior condizionale esatta) azzererebbe la varianza dei pesi ma è in generale intrattabile — lo stesso problema che motiva i filtri approssimati come EKF e UKF. La proposta più usata in pratica è la prior di transizione $q = p(\mathbf{z}_t \mid \mathbf{z}_{t-1})$, semplice ma spesso inefficiente quando le osservazioni sono informative.

Modelli di Spazio degli Stati Neurali

Reti neurali come modelli di transizione e osservazione — KVAE, RSSM, il World Model di Dreamer e l'integrazione con l'apprendimento per rinforzo.

347.1 · La Fusione tra Modelli di Stato e Deep Learning

I modelli di spazio degli stati classici — Kalman, HMM, filtri particellari — hanno il pregio della interpretabilità e delle garanzie teoriche di convergenza, ma soffrono di una limitazione fondamentale: richiedono che le funzioni di transizione e osservazione siano specificate a priori o abbiano struttura parametrica molto semplice (lineare, gaussiana mista). I modelli di spazio degli stati neurali sostituiscono queste funzioni con reti neurali, guadagnando espressività a spese della trattabilità: se f_θ è una rete, la distribuzione posteriore $p(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T})$ non ha più forma analitica e richiede inferenza variazionale o campionamento. Il **Kalman Variational Autoencoder (KVAE)** (Fraccaro et al. 2017) è tra i primi modelli a combinare la struttura lineare-gaussiana del filtro di Kalman con encoder e decoder neurali: le variabili latenti si dividono in "latenti locali" (a_t) processate dal filtro di Kalman e "latenti globali" (z_t) modellate con VAE, permettendo di sfruttare la struttura sequenziale senza rinunciare alla ricchezza delle rappresentazioni neurali.

Il **Recurrent State Space Model (RSSM)** (Hafner et al. 2019), alla base del modello Dreamer per il reinforcement learning model-based, separa lo spazio degli stati in una componente deterministica $\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{z}_{t-1}, \mathbf{a}_{t-1})$ — aggiornata da una RNN — e una componente stocastica $\mathbf{z}_t \sim p_\phi(\mathbf{z}_t \mid \mathbf{h}_t)$ — variabile latente gaussiana. Questa separazione permette di fare planning immaginando traiettorie nello spazio latente senza interagire con l'ambiente reale: l'agente di Dreamer apprende una policy ottimizzando i reward previsti dal world model neurale, riducendo di ordini di grandezza il numero di interazioni ambientali necessarie rispetto ai metodi di RL model-free. La chiave teorica di questa architettura è che la componente deterministica $\mathbf{h}_t$ fornisce il "contesto" per la transizione stocastica, permettendo al modello di catturare dipendenze a lungo termine che una variabile di stato puramente markoviana non potrebbe rappresentare.

347.2 · Dreamer: Planning nell'Immaginazione con World Models

Il sistema **Dreamer** (Hafner, Lillicrap, Ba e Norouzi, ICLR 2020) ha dimostrato che un agente può apprendere comportamenti complessi quasi interamente attraverso la *fantasia* — traiettorie simulate nello spazio latente del world model — riducendo le interazioni ambientali reali di uno o due ordini di grandezza rispetto ai metodi model-free come PPO. L'architettura di Dreamer

si articola in tre componenti che si addestrano congiuntamente. Il **world model** (RSSM) apprende la dinamica dell'ambiente dallo stream di esperienze reali minimizzando una loss ELBO che bilancia la ricostruzione delle osservazioni, la previsione dei reward, e la regolarizzazione KL della distribuzione stocastica; i **comportamenti latenti** (actor-critic) vengono addestrati interamente su traiettorie generate dal world model attraverso differenziazione attraverso il tempo — poiché il world model è differenziabile rispetto allo stato e all'azione, i gradienti delle reward future possono essere propagati all'indietro attraverso le transizioni latenti fino alla policy, senza il rumore tipico del policy gradient stocastico; la **stima del valore** opera sugli stati latenti del world model, apprendendo quante reward future ci si aspetta da ogni stato immaginato. **DreamerV3** (Hafner et al. 2023) ha esteso questo framework con tecniche di normalizzazione che rendono lo stesso set di iperparametri applicabile a domini eterogenei — da Atari ai giochi 3D di DeepMind Control a Minecraft — senza alcun adattamento, dimostrando la robustezza del world model come substrato per l'apprendimento di comportamenti generalisti.

Reti Bayesiane Dinamiche e Modelli Grafici Temporali

Estensione delle Bayesian Networks al dominio temporale — struttura a due livelli, apprendimento della struttura dinamica e applicazioni alla modellazione di serie storiche.

381.1 · La Struttura a Due Livelli delle DBN

Le **Reti Bayesiane Dinamiche (DBN)** (Dean e Kanazawa 1989) generalizzano sia le Bayesian Networks statiche che i modelli HMM e Kalman al caso di sistemi con più variabili che evolvono nel tempo. Una DBN è specificata da due componenti: la *distribuzione iniziale* $P(\mathbf{z}_1)$ sulle variabili al tempo 1, e il *modello di transizione a due livelli* (2-TBN) che specifica come le variabili al tempo t dipendono da quelle al tempo $t-1$ tramite un DAG che coinvolge variabili di entrambi i livelli temporali. Il modello congiunto si ottiene replicando il 2-TBN per tutti i passi temporali: $P(\mathbf{z}_{1:T}) = P(\mathbf{z}_1)\prod_{t=2}^{T} P(\mathbf{z}_t|\mathbf{z}_{t-1})$ dove $P(\mathbf{z}_t|\mathbf{z}_{t-1})$ si fattorizza secondo il DAG del 2-TBN. Questa struttura unifica i modelli visti in precedenza: l'HMM è una DBN con un'unica variabile latente discreta; il modello di Kalman è una DBN con variabili latenti continue gaussiane e transizioni lineari; i modelli di spazio degli stati neurali (§347) sono DBN con transizioni parametrizzate da reti neurali.

L'inferenza esatta nelle DBN utilizza l'**algoritmo di interfaccia** (Boyen e Koller 1998): invece di propagare la distribuzione completa sullo stato al tempo t — che può avere dimensione esponenziale nel numero di variabili latenti — si mantiene una distribuzione fattorizzata

sull'interfaccia, cioè le variabili che connettono il livello t al livello $t+1$. Questa approssimazione, nota come BK approximation, riduce il costo di ogni passo di filtering da esponenziale a polinomiale nel numero di variabili, al costo di introdurre un bias di approssimazione che dipende dalla forza delle correlazioni tra le variabili di interfaccia. Le DBN con variabili continue e interazioni non lineari tra variabili richiedono metodi come il filtro particellare a rete (Koller e Lerner 2001) — un'estensione del particle filter (§346) che propaga particelle per le variabili latenti mentre mantiene la struttura a rete per sfruttare le indipendenze condizionali.

381.2 · Apprendimento della Struttura Causale nelle DBN

Una delle applicazioni più importanti delle DBN è l'apprendimento di relazioni causali in serie storiche — un problema complementare al causal discovery statico (§372–375) ma con la difficoltà aggiuntiva del dominio temporale. Il concetto di **causalità di Granger** (Granger 1969) formalizza la causalità temporale in modo operativo: la variabile X causa di Granger la variabile Y se la conoscenza del passato di X migliora la previsione del futuro di Y rispetto alla sola conoscenza del passato di Y. In pratica, si stima un modello VAR (Vector Autoregression) $\mathbf{y}_t = \sum_{\ell=1}^{L} A_\ell \mathbf{y}_{t-\ell} + \boldsymbol{\varepsilon}_t$ e si testa se i coefficienti della matrice A_ℓ corrispondenti alla variabile X siano significativamente diversi da zero nella regressione per Y. La causalità di Granger è una forma di predittibilità incrementale, non di causalità nel senso di Pearl — due variabili possono essere correlate nel tempo senza relazione causale diretta se sono entrambe causate da una terza variabile con ritardi diversi — ma rimane lo strumento più usato per l'analisi delle relazioni temporali nelle neuroscienze computazionali, nell'econometria, e nell'analisi delle reti biologiche.

Apprendimento di Sequenze con LSTM e Trasformers: Confronto Teorico

Memoria selettiva nelle RNN — la derivazione degli aggiornamenti LSTM, il problema del gradiente che svanisce, e perché i Transformer hanno superato le RNN sui task lunghi.

382.1 · Il Problema del Gradiente che Svanisce nelle RNN

Le reti ricorrenti standard aggiornano il loro stato nascosto con la regola $\mathbf{h}_t = \sigma(W_h \mathbf{h}_{t-1} + W_x \mathbf{x}_t + \mathbf{b})$, e il gradiente rispetto ai parametri si propaga all'indietro nel tempo tramite la regola della catena: $\partial \mathbf{h}_t / \partial \mathbf{h}_{t-k} = \prod_{i=1}^{k} W_h^T \operatorname{diag}(\sigma'(\mathbf{a}_{t-i+1}))$. Questo prodotto di matrici tende a zero esponenzialmente se il raggio spettrale di W_h è minore di uno — il **problema del gradiente che svanisce** (Hochreiter 1991, Bengio et al. 1994) — oppure a divergere se è

maggiore di uno — il problema del gradiente esplosivo. In entrambi i casi, la rete non può apprendere dipendenze a lungo raggio perché il segnale di gradiente che "collega" errori lontani nel tempo diventa inutilizzabile. La soluzione standard è il **gradient clipping** per l'esplosione (limitare la norma del gradiente a una soglia massima), ma il vanishing è più problematico perché non può essere semplicemente "clippato".

382.2 · LSTM: Porte Differenziabili come Memoria Selettiva

Le **Long Short-Term Memory (LSTM)** (Hochreiter e Schmidhuber 1997) risolvono il vanishing gradient con un'architettura che mantiene uno stato di cella $\mathbf{c}_t$ che si aggiorna tramite addizioni invece di moltiplicazioni ricorrenti — garantendo che i gradienti possano fluire indietro nel tempo senza attenuarsi. Le quattro porte dell'LSTM controllano selettivamente questo flusso:

$$\mathbf{f}_t = \sigma(W_f[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f) \quad \text{(porta di dimenticanza)}$$

$$\mathbf{i}_t = \sigma(W_i[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i) \quad \text{(porta di input)}$$

$$\tilde{\mathbf{c}}_t = \tanh(W_c[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_c) \quad \text{(contenuto candidato)}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \quad \text{(aggiornamento stato cella)}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \quad \text{(output filtrato)}$$

La porta di dimenticanza $\mathbf{f}_t$ decide quale informazione della cella precedente mantenere (valori vicini a 1) o eliminare (valori vicini a 0); la porta di input decide quale parte del nuovo contenuto aggiungere alla cella; la porta di output filtra quale parte della cella esporre come stato nascosto. La chiave del successo dell'LSTM contro il vanishing gradient è l'aggiornamento additivo della

cella: il gradiente rispetto a $\mathbf{c}_{t-1}$ è $\partial \mathbf{c}_t / \partial \mathbf{c}_{t-1} = \text{diag}(\mathbf{f}_t)$ — il prodotto di porte di dimenticanza lungo il tempo, che se vicine a 1 mantiene il gradiente quasi intatto per lunghe sequenze. Il GRU (Gated Recurrent Unit, Cho et al. 2014) semplifica l'LSTM unendo la porta di dimenticanza e quella di input in una singola "porta di aggiornamento", con performance simile e minore costo computazionale.

382.3 · Perché i Transformer Superano le RNN su Sequenze Lunghe

Nonostante l'ingegnosità dell'LSTM, i Transformer (Vaswani et al. 2017) hanno soppiantato le RNN sulla maggior parte dei task di sequenza a partire dal 2018. La ragione è strutturale: il meccanismo di attenzione dell'auto-attenzione crea connessioni dirette tra qualsiasi coppia di posizioni nella sequenza con costo $O(1)$ di "distanza di segnale" — il gradiente scorre direttamente tra posizioni lontane senza attraversare nodi intermedi. Nelle RNN, la distanza di segnale tra le posizioni i e j è $O(|i-j|)$ applicazioni della matrice ricorrente, anche con l'LSTM. La tabella di confronto illustra il tradeoff:

Proprietà	**LSTM**	**Transformer**
Complessità per layer	$O(nd^2)$	$O(n^2 d)$
Operazioni sequenziali	$O(n)$	$O(1)$
Distanza di segnale max.	$O(n)$	$O(1)$
Memoria	$O(n)$	$O(n^2)$
Parallelizzabile in training	No	Sì
Adatto a sequenze molto lunghe ($n \gg d$)	Sì	No (senza sparse att.)

Il Transformer è parallelizzabile su hardware accelerato (GPU/TPU) perché non ha dipendenze sequenziali: tutti i token possono essere processati in parallelo nello stesso layer. L'LSTM deve processare i token nell'ordine della sequenza, il che lo rende intrinsecamente sequenziale e non parallelizzabile nella dimensione temporale. Questo vantaggio computazionale, unito alla minore distanza di segnale, spiega il dominio empirico dei Transformer — e la recente famiglia dei modelli a stato strutturato (SSM: Mamba, H3, Hyena) tenta di combinare il meglio dei due mondi: complessità lineare in n come le RNN e parallelismo in training come i Transformer.

Modelli Autoregressivi e la Stima della Densità

La regola della catena come architettura generativa — NADE, PixelCNN, Masked Autoencoder e la modellazione esplicita della densità.

383.1 · La Fattorizzazione Autoregressiva della Densità

Qualunque distribuzione congiunta può essere fattorizzata come prodotto di condizionali tramite la regola della catena della probabilità: $P(\mathbf{x}) = \prod_{i=1}^{d} P(x_i | x_1, \ldots, x_{i-1})$. I **modelli autoregressivi** sfruttano questa fattorizzazione parametrizzando ogni condizionale con una rete neurale che prende in input tutte le variabili precedenti: $P(\mathbf{x}; \theta) = \prod_{i=1}^{d} P_{\theta_i}(x_i | x_{<i})$. Il vantaggio fondamentale è la calcolabilità esatta della densità: non c'è approssimazione, non c'è lower bound (come nell'ELBO dei VAE), non c'è stima di normalizzazione (come nei modelli energetici). Questo rende i modelli autoregressivi ideali per la valutazione quantitativa della qualità del modello in termini di log-likelihood sui dati di test — la metrica più diretta per la modellazione della densità. Il prezzo è la generazione sequenziale: per campionare $\mathbf{x}$, si devono campionare x_1, poi x_2 condizionato a x_1, poi x_3 condizionato a (x_1, x_2), e così via — rendendo la generazione $O(d)$ volte più lenta dell'inferenza parallela dei VAE o dei diffusion models (che generano tutti i pixel/token in parallelo in ciascuno step di denoising).

Il **Neural Autoregressive Distribution Estimator (NADE)** (Larochelle e Murray 2011) è stato tra i primi modelli autoregressivi scalabili: parametrizza ogni condizionale con un layer nascosto condiviso che può essere calcolato incrementalmente, riducendo il costo da $O(d^2)$ a $O(d \cdot h)$ dove h è la dimensione del layer nascosto. **PixelCNN** (van den Oord et al. 2016) estende l'idea alle immagini tramite convoluzioni causali mascherate: ogni pixel è condizionato a tutti i pixel precedenti nell'ordine raster (da sinistra a destra, dall'alto in basso), con una maschera nella convoluzione che garantisce l'assenza di connessioni da pixel futuri a pixel passati. PixelCNN++ (Salimans et al. 2017) aggiunge discretizzazione logistica mixta per l'output, connessioni skip, e dropout, raggiungendo log-likelihood vicine allo stato dell'arte su CIFAR-10 come modello esplicito di densità.

383.2 · Masked Autoencoders e il Collegamento con BERT

Il **Masked Autoencoder for Distribution Estimation (MADE)** (Germain et al. 2015) applica la fattorizzazione autoregressiva agli autoencoder standard tramite una maschera che garantisce l'ordine autoregressivo: si assegnano numeri d'ordine a ogni unità della rete, e le connessioni da unità con numero alto a unità con numero basso vengono azzerate tramite una maschera moltiplicata ai pesi. Il risultato è un autoencoder che implementa la fattorizzazione autoregressiva in una sola passata forward, invece delle d passate separate necessarie nei modelli autoregressivi naïve. La connessione con BERT (Devlin et al. 2019) e i modelli linguistici a maschera è diretta: BERT pre-addestra una rete Transformer sullo stesso obiettivo del MADE — predire i token mascherati dato il contesto non mascherato — ma con due differenze fondamentali. Prima differenza: BERT usa un approccio non autoregressivo (tutti i token mascherati

vengono predetti in parallelo, non sequenzialmente) il che velocizza il training ma introduce correlazioni spurie tra le predizioni dei token mascherati. Seconda differenza: BERT usa la bidirezionalità — ogni token può guardare sia i token precedenti che successivi — mentre MADE è strettamente causale. Il Masked Autoregressive Flow (MAF, Papamakarios et al. 2017) usa MADE come layer base di un Normalizing Flow, combinando la trattabilità della densità esatta con la potenza dei flussi normalizzanti.

Apprendimento Contrastivo e la Struttura della Rappresentazione

Imparare rappresentazioni senza etichette tramite confronto — InfoNCE, SimCLR, CLIP e la geometria dell'embedding spazio.

384.1 · La Loss InfoNCE come Stima della Mutua Informazione

L'**apprendimento contrastivo** è il paradigma dominante per l'apprendimento non supervisionato di rappresentazioni ed è alla base di modelli come SimCLR, MoCo, BYOL e CLIP. L'idea fondamentale è di fare in modo che le rappresentazioni di due "viste" dello stesso dato siano simili (esempi positivi), mentre le rappresentazioni di dati diversi siano distanti (esempi negativi). La **loss InfoNCE** (Oord, Li e Vinyals 2018), originariamente derivata nel contesto di CPC (Contrastive Predictive Coding), formalizza questo obiettivo come:

$$\mathcal{L}_{\text{InfoNCE}} = -\mathbb{E}\left[\log \frac{e^{f(\mathbf{x},\mathbf{x}^+)/\tau}}{\sum_{j=0}^{N} e^{f(\mathbf{x},\mathbf{x}_j)/\tau}}\right]$$

dove $\mathbf{x}^+$ è la vista positiva (stesso dato, aumentazione diversa), $\mathbf{x}_j$ sono N esempi negativi (dati diversi), $f(\mathbf{x}, \mathbf{x}')$ è la funzione di similarità (tipicamente il prodotto interno normalizzato dei vettori di embedding), e $\tau > 0$ è la temperatura che controlla la "sharpness" della distribuzione. La loss InfoNCE è un lower bound alla mutua informazione $I(\mathbf{x}; \mathbf{x}^+)$ con $\log N$ di gap: $I(\mathbf{x}; \mathbf{x}^+) \geq \log N - \mathcal{L}_{\text{InfoNCE}}$. Minimizzare la loss equivale quindi a massimizzare un lower bound alla mutua informazione tra le due viste — le rappresentazioni apprendono a catturare le informazioni condivise tra le augmentazioni dello stesso dato, che per costruzione corrispondono alle caratteristiche semantiche invarianti alle trasformazioni.

384.2 · SimCLR, MoCo, BYOL: la Triade dell'Apprendimento Contrastivo Visivo

SimCLR (Chen, Kornblith, Norouzi e Hinton 2020) ha dimostrato che l'apprendimento contrastivo con batch grandi ($N = 4096$ o più) e una testa di proiezione MLP (aggiunta sopra l'encoder ResNet ma scartata a fine training) raggiunge prestazioni paragonabili al training supervisionato su ImageNet con il 1% delle etichette. I fattori chiave del successo di SimCLR sono: le augmentazioni forti (crop casuale + color jitter + grayscale + blur), la dimensione del batch (che aumenta il numero di negativi), e soprattutto la testa di proiezione — rimuoverla dopo il training e usare solo l'encoder migliora significativamente le prestazioni sul downstream task, suggerendo che le rappresentazioni ottimali per il contrastive loss sono diverse da quelle ottimali per il transfer. **MoCo** (He, Fan, Wu, Xie e Girshick 2020) risolve il problema della necessità di batch enormi tramite una coda di negativi e un encoder momentum — un encoder separato i cui pesi sono l'EMA dell'encoder principale, che produce rappresentazioni dei negativi più stabili. **BYOL** (Bootstrap Your Own Latent, Grill et al. 2020) ha sorpreso la community dimostrando che i negativi possono essere eliminati completamente: due reti asimmetriche (online e target, con target aggiornato via EMA) che si predicono a vicenda convergono a buone rappresentazioni senza collasso verso soluzioni banali, grazie alla combinazione del predictor MLP e dell'EMA che rompe la simmetria tra le due reti. CLIP (Contrastive Language-Image Pre-training, Radford et al. 2021) estende il contrastive learning al dominio multimodale: testo e immagine vengono mappati in uno spazio condiviso tramite due encoder separati, con la loss InfoNCE calcolata sui 400 milioni di coppie testo-immagine del web. Il risultato è un modello con capacità di zero-shot classification straordinarie — CLIP raggiunge la accuratezza di un ResNet-50 supervisionato su ImageNet senza mai vedere le classi di ImageNet durante il training.

Processi Decisionali di Markov e l'Equazione di Bellman

La formalizzazione matematica del problema del controllo sequenziale — stati, azioni, reward, politiche e il principio di ottimalità di Bellman.

348.1 · Il Framework del Processo Decisionale di Markov

La teoria dell'apprendimento per rinforzo — il termine italiano è preciso e preferibile al calco diretto dall'inglese — si fonda sul framework del **Processo Decisionale di Markov (MDP)**, introdotto da Bellman (1957) sulla base del lavoro precedente di Howard sulla programmazione dinamica. Un MDP è una tupla $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$ dove $\mathcal{S}$ è lo spazio degli stati, $\mathcal{A}$ lo spazio delle

azioni, $P(s' \mid s, a)$ la funzione di transizione (la probabilità di passare allo stato s' eseguendo l'azione a nello stato s), $R(s, a)$ la funzione di ricompensa, e $\gamma \in [0, 1)$ il fattore di sconto che pesa le ricompense future rispetto a quelle immediate. Un agente interagisce con l'MDP seguendo una **politica** $\pi(a \mid s)$ — una distribuzione sulle azioni condizionata allo stato corrente — e l'obiettivo è trovare la politica ottimale π^* che massimizza il *ritorno atteso scontato* $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$.

La quantità centrale dell'RL è la **funzione valore** $V^\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]$ — il ritorno atteso a partire dallo stato s seguendo la politica π — e la **funzione Q** (o funzione azione-valore) $Q^\pi(s, a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$ — il ritorno atteso eseguendo l'azione a nello stato s e poi seguendo π. La politica ottimale soddisfa $\pi^*(a \mid s) = \mathbf{1}[a = \arg\max_{a'} Q^*(s, a')]$ dove Q^* è la funzione Q ottimale. Il legame tra V^π, Q^π e π è catturato dall'**equazione di Bellman**:

$$V^\pi(s) = \sum_a \pi(a \mid s) \sum_{s'} P(s' \mid s, a) \left[R(s, a) + \gamma V^\pi(s')\right]$$

Questa equazione esprime una condizione di autoconsistenza: il valore di uno stato deve essere uguale alla ricompensa immediata attesa più il valore scontato degli stati successori. Per la politica ottimale, si ha l'**equazione di Bellman ottimale** $V^*(s) = \max_a \sum_{s'} P(s' \mid s, a)[R(s, a) + \gamma V^*(s')]$ — un'equazione di punto fisso nella funzione valore ottimale. L'esistenza e l'unicità di V^* sono garantite dal fatto che l'operatore di Bellman ottimale $\mathcal{T}^*[V](s) = \max_a \sum_{s'} P(s' \mid s, a)[R + \gamma V(s')]$ è una contrazione rispetto alla norma infinita con costante di Lipschitz $\gamma < 1$ — la condizione del Teorema del punto fisso di Banach.

348.2 · Programmazione Dinamica: Policy Iteration e Value Iteration

Quando il modello dell'ambiente — cioè la funzione di transizione P e la funzione di ricompensa R — è noto, il problema di trovare la politica ottimale si riduce alla risoluzione dell'equazione di Bellman, che può essere affrontata con due algoritmi classici di programmazione dinamica. La **Policy Iteration** (Howard 1960) alterna due passi: la *valutazione della politica* (calcolo di V^π per la politica corrente tramite la soluzione del sistema lineare $V = R^\pi + \gamma P^\pi V$) e il *miglioramento della politica* (aggiornamento della politica tramite l'argmax della funzione Q corrente). La convergenza è garantita: ogni iterazione produce una politica non peggiore della precedente, e la sequenza converge alla politica ottimale in un numero finito di passi su MDP finiti. La **Value Iteration** (Bellman 1957) è più semplice: applica ripetutamente l'operatore di Bellman ottimale $V_{k+1} = \mathcal{T}^*[V_k]$ fino alla convergenza. Poiché $\mathcal{T}^*$ è una contrazione con fattore γ, la convergenza è geometrica: $\|V_k - V^*\|_\infty \leq \gamma^k \|V_0 - V^*\|_\infty$. Il limite di questi metodi è la

necessità di conoscere P e R esplicitamente — il cosiddetto *modello dell'ambiente* — che è quasi sempre ignoto nelle applicazioni reali dove l'agente deve apprendere attraverso l'interazione.

353.2 · UCRL2 e gli Algoritmi Ottimistici Continui

L'algoritmo RMAX è concettualmente semplice ma applicabile solo a MDP con spazi di stati e azioni finiti e piccoli. L'**UCRL2** (Jaksch, Ortner e Auer 2010) estende il principio ottimistico al setting online su MDP con spazi finiti senza assumere che il modello venga stimato a sufficienza prima di usarlo: ad ogni episodio di lunghezza au_k, si costruisce un set di confidenza $\mathcal{M}_k$ dei modelli MDP compatibili con i dati osservati, e si seleziona il modello ottimisticamente migliore all'interno di quel set tramite Extended Value Iteration. Il regret cumulativo di UCRL2 dopo T passi su un MDP con S stati, A azioni e diametro D soddisfa $R(T) \leq 34\, DS\sqrt{AT\log(T/\delta)}$ con probabilità almeno $1-\delta$ — un bound che scala come $\sqrt{T}$, lo stesso dei bandit (§339), confermando che la sample complexity dell'RL è fondamentalmente più difficile dei problemi di supervisione diretta. Il problema del **reward-free RL** (Jin, Krishnamurthy, Simchowitz e Yu 2020) separa ulteriormente la fase di esplorazione da quella di pianificazione: nella fase di esplorazione l'agente non riceve alcun reward e raccoglie dati sull'ambiente; nella fase di pianificazione viene specificato un reward, e l'agente deve produrre una politica ottimale usando solo i dati raccolti. Questo setting modella accuratamente il pre-training senza reward dei sistemi di RL reali (come l'esplorazione del mondo in Dreamer), e la sua sample complexity è $ilde{O}(S^2A/\varepsilon^2)$ per la garanzia ε-ottimale — confrontabile con RMAX ma con separazione netta tra esplorazione e sfruttamento.

Q-Learning: Apprendimento senza Modello con Garanzie di Convergenza

La stima model-free della funzione valore ottimale — aggiornamento TD, tabellare vs. approssimato, Deep Q-Network.

349.1 · La Differenza Temporale e l'Aggiornamento Q-Learning

Quando il modello dell'ambiente non è noto, l'agente deve stimare la funzione valore direttamente dall'esperienza. Il **Temporal Difference (TD) learning** (Sutton 1988) è l'idea chiave che ha reso possibile l'apprendimento per rinforzo scalabile: invece di aspettare la fine dell'episodio per aggiornare le stime (come nel Monte Carlo), si aggiorna la stima di $V(s_t)$ dopo ogni transizione usando come target la somma della ricompensa immediata e della stima corrente del valore del passo successivo — il cosiddetto *bootstrapping*. L'aggiornamento

TD(0) è: $V(s_t) \leftarrow V(s_t) + \alpha[R_{t+1} + \gamma V(s_{t+1}) - V(s_t)]$ dove il termine in parentesi — $\delta_t = R_{t+1} + \gamma V(s_{t+1}) - V(s_t)$ — è l'*errore di differenza temporale*. Il **Q-Learning** (Watkins 1989, con analisi di convergenza di Watkins e Dayan 1992) applica lo stesso principio alla funzione Q con un aggiornamento *off-policy*: indipendentemente dalla politica usata per generare i dati (la *politica comportamentale*), si aggiorna la funzione Q verso il target ottimale:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[R_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \right]$$

Il termine $\max_{a'} Q(s_{t+1}, a')$ è il target di Bellman ottimale: si assume che nel passo successivo si agirà ottimamente, indipendentemente da come si è in realtà comportati. Questa proprietà off-policy è cruciale: permette di imparare la politica ottimale mentre si esplora con una politica esploratoria (tipicamente ε-greedy, che sceglie l'azione migliore con probabilità $1-\varepsilon$ e un'azione casuale con probabilità ε). La convergenza del Q-learning tabellare (dove ogni coppia stato-azione ha la sua entry nella tabella Q) alla funzione Q ottimale è garantita sotto le condizioni di visita sufficientemente frequente di ogni coppia stato-azione e di un programma di learning rate che soddisfa le condizioni di Robbins-Monro.

349.2 · Deep Q-Network: Approssimazione della Funzione Q con Reti Neurali

L'approccio tabellare è applicabile solo a spazi stati-azione piccoli e discreti. Quando lo spazio degli stati è continuo o molto grande (come i pixel di un videogioco), si approssima la funzione Q con una rete neurale $Q_\theta(s, a)$ — il **Deep Q-Network (DQN)** di Mnih et al. (DeepMind, Nature 2015). La chiave del successo di DQN rispetto ai precedenti tentativi di approssimare la funzione Q con reti neurali sta in due innovazioni tecniche. La prima è il **replay buffer**: le esperienze (s_t, a_t, r_t, s_{t+1}) vengono memorizzate in un buffer di dimensione fissa, e ad ogni aggiornamento si campiona un mini-batch casuale dal buffer anziché usare le esperienze più recenti. Questo rompe la correlazione temporale tra esempi di training successivi — una violazione grave dell'assunzione di indipendenza degli esempi di SGD — e riduce la varianza degli aggiornamenti. La seconda innovazione è la **rete target**: si mantiene una copia separata della rete con pesi θ^- aggiornata solo ogni C passi, e si usa questa rete per calcolare il target $y_t = r_t + \gamma \max_{a'} Q_{\theta^-}(s_{t+1}, a')$. Senza la rete target, i pesi θ inseguirebbero un obiettivo che si sposta ad ogni aggiornamento — una forma di feedback positivo che porta all'instabilità del training.

Policy Gradient: REINFORCE e il Teorema del Gradiente della Politica

Ottimizzazione diretta della politica — il teorema fondamentale, REINFORCE con baseline, e la connessione con il BBVI.

350.1 · Il Gradiente dell'Obiettivo rispetto alla Politica

I metodi basati sulla funzione valore — Q-learning, TD — apprendono indirettamente una politica deterministica come argmax della funzione Q. I **metodi di gradiente della politica** rappresentano un paradigma alternativo: si parametrizza direttamente la politica $\pi_\theta(a \mid s)$ come una distribuzione sulle azioni, e si ottimizza l'obiettivo $J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[G(\tau)]$ — il ritorno atteso sulle traiettorie $\tau = (s_0, a_0, r_1, s_1, a_1, \ldots)$ generate dalla politica — tramite gradient ascent. Il calcolo di $\nabla_\theta J(\theta)$ non è immediato poiché l'obiettivo è un'aspettativa rispetto a una distribuzione che dipende da θ: non si può portare il gradiente dentro la sommatoria senza attenzione. Il **Teorema del Gradiente della Politica** (Williams 1992, Sutton et al. 2000) risolve il problema con il log-derivative trick già visto nella §282 per BBVI:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[G(\tau)\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t \mid s_t)\right]$$

La formula ha un'interpretazione intuitiva molto bella: il gradiente dell'obiettivo è l'aspettativa del prodotto tra il ritorno totale $G(\tau)$ e il gradiente del logaritmo della politica — il cosiddetto *score della politica*. Le traiettorie con ritorno alto alzano la probabilità delle azioni che le hanno generate (il gradiente di log-politica punta nella direzione di maggiore probabilità), mentre le traiettorie con ritorno basso la abbassano. L'algoritmo **REINFORCE** (Williams 1992) stima questo gradiente con un singolo campione di traiettoria, producendo uno stimatore non-biasato ma con varianza tipicamente molto elevata — uno dei problemi principali dell'RL basato su gradiente della politica.

350.2 · La Baseline e la Riduzione della Varianza

La varianza di REINFORCE può essere ridotta sottraendo una *baseline* $b(s_t)$ dal ritorno: $\nabla_\theta J(\theta) = \mathbb{E}[(G(\tau) - b(s_t))\sum_t \nabla_\theta \log \pi_\theta(a_t \mid s_t)]$. La baseline non introduce bias purché non dipenda dall'azione a_t: $\mathbb{E}[b(s_t)\nabla_\theta \log \pi_\theta(a_t \mid s_t)] = b(s_t)\nabla_\theta \sum_a \pi_\theta(a \mid s_t) = b(s_t)\nabla_\theta 1 = 0$. La scelta ottimale della baseline che minimizza la varianza dello stimatore è $b^*(s_t) = \mathbb{E}[G(\tau)(\nabla_\theta \log \pi_\theta)^2]/\mathbb{E}[(\nabla_\theta \log \pi_\theta)^2]$ — complessa da calcolare in pratica. La baseline più comune è la funzione valore $V_\phi(s_t)$ stimata separatamente: in tal caso, $G(\tau) - V_\phi(s_t)$ è chiamato

vantaggio $A(s_t, a_t)$ — la differenza tra il ritorno effettivo e il ritorno atteso per lo stato corrente. Questo porta naturalmente alle architetture Actor-Critic della §351.

350.2 · Generalized Advantage Estimation e il Tradeoff Bias-Varianza

La funzione di vantaggio $A(s_t, a_t) = Q(s_t, a_t) - V(s_t)$ misura quanto un'azione sia migliore o peggiore della media nello stato corrente, e è la quantità con la più bassa varianza tra i segnali di credit assignment disponibili. Nella pratica, A deve essere stimato: una stima a n passi $\hat{A}_t^{(n)} = \sum_{k=0}^{n-1} \gamma^k r_{t+k} + \gamma^n V_\phi(s_{t+n}) - V_\phi(s_t)$ introduce un tradeoff tra bias (bootstrapping da V_ϕ, che può essere impreciso) e varianza (ritorno episodico completo, molto rumoroso). Il **Generalized Advantage Estimation (GAE)** (Schulman et al. 2016) risolve elegantemente questo tradeoff come combinazione esponenzialmente pesata degli stimatori a tutti i passi n: $\hat{A}_t^{mGAE}(\gamma, \lambda) = \sum_{k=0}^{\infty} (\gamma\lambda)^k \delta_{t+k}$ dove $\delta_t = r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$ è l'errore TD a un passo. Il parametro $\lambda \in [0, 1]$ controlla il tradeoff: $\lambda = 0$ riduce a TD(0) (puro bootstrapping, bassa varianza, alto bias), $\lambda = 1$ riduce al ritorno episodico completo scontato (nessun bootstrapping, alta varianza, nessun bias sotto V_ϕ perfetta). GAE è l'estimatore del vantaggio standard in PPO e quasi tutti i metodi actor-critic moderni, ed è implementato come semplice prodotto ricorrente sui delta TD in ordine inverso del tempo, con costo $O(T)$ per episodio.

350.3 · La Connessione tra Policy Gradient e ELBO Variazionale

La struttura matematica del policy gradient e dell'inferenza variazionale (§282) sono sorprendentemente simili. Entrambi usano il log-derivative trick per calcolare il gradiente di un'aspettativa rispetto a parametri della distribuzione campionante; entrambi soffrono di alta varianza degli stimatori; entrambi usano baseline (REINFORCE) o la reparametrizzazione (VAE) per ridurre la varianza. Questa connessione non è casuale: il policy gradient è letteralmente un REINFORCE applicato alla distribuzione delle traiettorie $p_h eta(au)$, e il reparameterization trick applicato alla policy stocastica corrisponde alla differenziazione deterministica attraverso campionatori di azioni continue. In effetti, il framework Control as Inference (Levine 2018) formalizza l'RL come un problema di inferenza variazionale su un modello probabilistico con variabili di ottimalità: la politica ottimale emerge come la distribuzione posteriore sulle azioni data l'osservazione di aver ottenuto reward elevata, e l'ottimizzazione dell'ELBO su questo modello corrisponde esattamente all'apprendimento per rinforzo con maximum entropy regularization $\mathcal{H}[\pi]$. Questo legame ha prodotto algoritmi come SAC (Soft Actor-Critic), che tratta la politica come una distribuzione da ottimizzare variazionalmente rispetto a un target a entropia massima, ottenendo stabilità notevolmente superiore a PPO su task di controllo continuo.

Architetture Actor-Critic e A3C

La combinazione di gradiente della politica e stima della funzione valore — il vantaggio, la sincronizzazione asincrona e la stabilità del training.

351.1 · Actor e Critic come Componenti Complementari

Le architetture **Actor-Critic** separano esplicitamente i due ruoli che nell'RL basato sulla funzione valore sono impliciti nella stessa struttura. L'*attore* (actor) è la politica $\pi_\theta(a \mid s)$ che produce azioni — il componente "decisionale". Il *critico* (critic) è la funzione valore $V_\phi(s)$ che valuta quanto sia buona la situazione corrente — il componente "valutativo". Ad ogni passo, l'attore sceglie un'azione, il critico stima il vantaggio $A(s_t, a_t) = r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$ (la differenza TD come stima del vantaggio), e si aggiornano entrambi: l'attore con il gradiente della politica pesato per il vantaggio, il critico minimizzando l'errore quadratico della funzione valore. Il vantaggio stimato con la differenza TD è una stima bilanciata tra la bassa varianza del bootstrapping (un singolo passo) e il basso bias del ritorno episodico completo: $A^{(n)}(s_t) = \sum_{k=0}^{n-1} \gamma^k r_{t+k} + \gamma^n V_\phi(s_{t+n}) - V_\phi(s_t)$ con n passi di lookahead. Il **GAE (Generalized Advantage Estimation, Schulman et al. 2016)** media questi estimatori con pesi esponenziali $(\gamma\lambda)^k$, producendo un tradeoff bias-varianza controllato dal parametro $\lambda \in [0, 1]$.

L'**Asynchronous Advantage Actor-Critic (A3C)** (Mnih et al. DeepMind 2016) ha risolto un problema pratico cruciale del training Actor-Critic: la correlazione temporale tra esperienze successive nella stessa traiettoria introduce instabilità nel training della rete, analogamente a quanto avviene nel Q-learning senza replay buffer. A3C usa n agenti paralleli che interagiscono con copie indipendenti dell'ambiente e accumulano gradienti su traiettorie separate, poi aggiornano asincronamente i pesi di una rete globale condivisa. La diversità delle traiettorie provenienti da ambienti diversi (spesso in stati diversi dell'ambiente, a volte anche con seed casuali diversi) de-correla le esperienze senza la necessità di un replay buffer esplicito. A3C è stato il primo metodo di RL ad addestrare agenti competitivi a livello umano su decine di videogiochi Atari usando solo la CPU, senza GPU, in poche ore di training.

351.2 · Soft Actor-Critic e il Maximum Entropy Framework

Il **Soft Actor-Critic (SAC)** (Haarnoja, Zhou, Abbeel e Levine 2018) risolve due problemi critici dell'Actor-Critic standard: la fragilità rispetto alla scelta dell'iperparametro del learning rate e la tendenza all'esplorazione insufficiente in ambienti continui. SAC ottimizza un obiettivo

di *maximum entropy RL*: invece di massimizzare solo la reward cumulativa, si massimizza la reward più l'entropia della policy ad ogni passo, $J(\pi) = \sum_t \mathbb{E}[r_t + \alpha\mathcal{H}(\pi(\cdot|s_t))]$, dove $\alpha > 0$ è il coefficiente di temperatura che bilancia il sfruttamento della reward con l'esplorazione tramite alta entropia. La policy ottimale in questo framework è $\pi^*(a|s) \propto \exp(Q^*(s,a)/\alpha)$ — una softmax della funzione Q invece dell'argmax deterministico del Q-learning standard. Questo induce esplorazione automatica senza meccanismi ε-greedy: la policy ad entropia massima sceglie azioni in proporzione alla loro Q-value stimata, campionando più uniformemente quando la funzione Q è piatta (situazioni poco conosciute) e concentrandosi sull'azione migliore quando la Q è fortemente peaked (situazioni ben apprese). SAC mantiene due critici paralleli (twin critics) per ridurre l'overestimation bias del Q-learning, e adatta automaticamente la temperatura α tramite un'equazione duale che mantiene l'entropia target: $\min_\alpha \mathbb{E}_{a\sim\pi}[-\alpha \log \pi(a|s_t) - \alpha\bar{\mathcal{H}}]$ dove $\bar{\mathcal{H}}$ è l'entropia target, tipicamente $-\dim(\mathcal{A})$. Il risultato è un algoritmo notevolmente stabile e robusto che supera sistematicamente PPO e DDPG su benchmark di robotica continua (MuJoCo, OpenAI Gym) con minore sensibilità agli iperparametri.

PPO e TRPO: Ottimizzazione della Politica con Vincoli di Monotonia

Aggiornamenti stabili della politica — la regione fiduciosa, il clipping surrogato e il successo pratico di PPO.

352.1 · Il Problema degli Aggiornamenti Eccessivi

Un problema fondamentale dei metodi di gradiente della politica è la *fragilità degli aggiornamenti*: un singolo passo di gradient ascent troppo grande può portare la politica in una regione dello spazio dei parametri dove la performance è drasticamente peggiore, da cui è difficile uscire perché le nuove traiettorie sono generate da una politica diversa e non informativa. A differenza del supervised learning — dove un aggiornamento sbagliato è "corretto" dal campione successivo perché la distribuzione dei dati non cambia — nell'RL la distribuzione dei dati dipende dalla politica stessa: se la politica peggiora, anche le traiettorie future peggioreranno, creando un circolo vizioso. Il **Trust Region Policy Optimization (TRPO)** (Schulman, Levine, Moritz, Jordan e Abbeel, ICML 2015) formalizza la soluzione: aggiornare la politica solo all'interno di una "regione fiduciosa" definita da una vincolo sulla divergenza KL tra la politica vecchia e quella nuova:

$$\max_\theta \ \mathbb{E}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} A_t\right] \quad \text{s.t.} \quad \mathbb{E}_t[\text{KL}(\pi_{\theta_{\text{old}}}(\cdot \mid s_t) \| \pi_\theta(\cdot \mid s_t))] \leq \delta$$

Il termine $\pi_\theta/\pi_{\theta_{\text{old}}}$ è il rapporto di importanza che corregge il fatto che i dati di training sono stati generati dalla vecchia politica anziché dalla nuova — una forma di off-policy correction analoga all'importance sampling del §338. TRPO risolve il problema vincolato tramite l'algoritmo del gradiente coniugato con le approssimazioni di Fisher-vector product per la matrice di curvatura KL, ottenendo una soluzione che garantisce la monotonia del miglioramento della politica per l'approssimazione al primo ordine dell'obiettivo surrogato.

352.2 · PPO: La Versione Pratica con Clipping

Nonostante le forti garanzie teoriche, TRPO è computazionalmente oneroso (il calcolo del Fisher-vector product richiede due backward pass per ogni aggiornamento) e difficile da implementare correttamente in presenza di architetture complesse. Il **Proximal Policy Optimization (PPO)** (Schulman, Wolski, Dhariwal, Radford e Klimov, OpenAI 2017) raggiunge prestazioni simili a TRPO con una soluzione molto più semplice: invece del vincolo KL esplicito, si usa una funzione obiettivo "clippata" che penalizza implicitamente i grandi cambiamenti della politica. La loss di PPO con clipping è:

$$\mathcal{L}^{\text{CLIP}}(\theta) = \mathbb{E}_t[\min(r_t(\theta)A_t,\ \text{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon)A_t)]$$

dove $r_t(\theta) = \pi_\theta(a_t \mid s_t)/\pi_{\theta_{\text{old}}}(a_t \mid s_t)$ è il rapporto di importanza e $\varepsilon \approx 0.2$ controlla l'ampiezza della regione consentita. Il meccanismo di clipping funziona come segue: quando il vantaggio è positivo, si incentiva ad aumentare la probabilità dell'azione, ma il beneficio si azzera non appena il rapporto supera $1+\varepsilon$; quando il vantaggio è negativo, si penalizza la probabilità dell'azione, ma la penalità si azzera quando il rapporto scende sotto $1-\varepsilon$. In entrambi i casi, la politica viene spinta verso i dati di training ma non oltre una distanza controllata. PPO è diventato l'algoritmo di RL di default in OpenAI e nella maggior parte dei laboratori di ricerca, ed è l'algoritmo usato nel training dell'RLHF per i modelli linguistici di grandi dimensioni — la fase che ha prodotto ChatGPT e i suoi successori.

Teoria dell'RL: PAC-MDP, RMAX e la Sample Complexity

Garanzie teoriche per l'esplorazione efficiente — la distinzione tra stati noti e ignoti, il bound di sample complexity e i limiti informativi.

353.1 · Il Problema dell'Esplorazione e la Sample Complexity

La teoria pratica dell'apprendimento per rinforzo si concentra su algoritmi che funzionano bene empiricamente, ma la teoria matematica affronta domande più fondamentali: quante interazioni con l'ambiente sono necessarie per trovare una politica ε-ottimale con alta probabilità? Questa è la **sample complexity dell'RL**, analoga alla sample complexity del supervised learning (PAC learning). Il risultato fondamentale è il framework **PAC-MDP** (Probably Approximately Correct for MDPs, Strehl e Littman 2008): un algoritmo è PAC-MDP se, con probabilità almeno $1-\delta$, produce una politica ε-ottimale dopo al più $\text{poly}(|\mathcal{S}|, |\mathcal{A}|, 1/\varepsilon, 1/\delta, 1/(1-\gamma))$ passi di esplorazione non-ottimale. L'algoritmo **RMAX** (Brafman e Tennenholtz 2002) è il più semplice a soddisfare questa proprietà: si mantiene una distinzione tra stati "noti" (visitati sufficientemente spesso da avere stime accurate di P e R) e stati "ignoti" (da esplorare). Per gli stati ignoti, si assegna un reward ottimistico $R_{\max}$ — il massimo reward possibile — incentivando l'agente a esplorare. Questa strategia di esplorazione ottimistica è la versione model-based del principio UCB dei banditi (§339): si tratta ogni stato ignoto come se fosse la migliore esperienza possibile, motivando la sua visita. RMAX trova una politica ε-ottimale in $O(|\mathcal{S}|^2|\mathcal{A}|/\varepsilon^3)$ iterazioni — polinomiale nel numero di stati e azioni, ma esponenziale in problemi con spazi degli stati continui o molto grandi, il che motiva le approssimazioni di funzione della §349.

Apprendimento PAC: la Struttura Matematica della Generalizzazione

Quando e quanto velocemente un algoritmo può imparare — errore empirico vs. vero, VC dimension e la dicotomia fondamentale del learning.

354.1 · Il Framework PAC di Valiant

Nel 1984, Leslie Valiant pubblicò un articolo intitolato *A Theory of the Learnable* che poneva per la prima volta in modo rigoroso la domanda: cos'è computazionalmente possibile imparare, e con quante risorse? Il framework **PAC (Probably Approximately Correct)** formalizza l'apprendimento come un problema di generalizzazione statisticamente fondato: dato un concetto target c^* in una classe di ipotesi $\mathcal{H}$, un dataset di m esempi i.i.d. da una distribuzione $\mathcal{D}$, e un algoritmo di apprendimento, si vuole garantire che con probabilità almeno $1-\delta$ (rispetto alla scelta casuale del dataset) l'ipotesi h restituita dall'algoritmo abbia errore vero $L_{\mathcal{D}}(h) = P_{\mathcal{D}}[h(x) \neq c^*(x)] \leq \varepsilon$. Un concetto è **PAC-apprendibile** se esiste un algoritmo che lo apprende in modo PAC con un numero di esempi $m = \text{poly}(1/\varepsilon, 1/\delta, \dim(\mathcal{H}))$ e in tempo computazionale polinomiale.

Il legame tra il numero di esempi necessari e la complessità della classe di ipotesi è catturato dalla **VC dimension** (Vapnik-Chervonenkis 1971). La VC dimension di una classe $\mathcal{H}$ è il massimo numero di punti che $\mathcal{H}$ può "rompere" — assegnare arbitrariamente a qualunque combinazione di etichette binarie. Per gli iperpiani in $\mathbb{R}^d$, la VC dimension è $d+1$; per le reti neurali con attivazione sigmoide, è polinomiale nel numero di pesi (Goldberg e Jerrum 1995, Bartlett et al. 2019). Il teorema di Vapnik-Chervonenkis afferma che con $m = O((d_{\text{VC}} \log(1/\varepsilon) + \log(1/\delta))/\varepsilon)$ esempi, qualunque algoritmo che azzera l'errore empirico generalizza con errore vero al più ε — il risultato fondamentale che quantifica il "quanto dati bastano".

354.2 · La VC Dimension delle Reti Neurali

Il calcolo della VC dimension per le reti neurali è un problema delicato che ha richiesto decenni per essere risolto soddisfacentemente. Per reti con attivazioni a soglia (Heaviside), la VC dimension è $\Theta(W \log W)$ dove W è il numero di parametri — un risultato di Goldberg e Jerrum (1993). Per reti con attivazioni continue come ReLU, i risultati più recenti di Bartlett, Harvey, Liaw e Mehrabian (2019) mostrano che la VC dimension è $O(WL \log W)$ dove L è il numero di layer, con un matching lower bound $\Omega(WL \log(W/L))$. Questi risultati sembrano suggerire che le reti neurali moderne — con miliardi di parametri — non possano generalizzare: il bound sulla generalizzazione sarebbe inutilmente lasco. Tuttavia, i risultati empirici mostrano il contrario: reti ampiamente sovra-parametrizzate generalizzano eccellentemente. Questo paradosso, noto come il *puzzle della generalizzazione nel deep learning*, è al centro della ricerca teorica attuale e motiva i bound di complessità più fini discussi nelle §355–358.

Rademacher Complexity: Complessità Dipendente dai Dati

Un misuratore di complessità più fine della VC dimension — definizione, proprietà di contrazione, bound data-dependent.

355.1 · Definizione e Interpretazione Intuitiva

La VC dimension è una misura di complessità *worst-case*: dipende solo dalla classe di ipotesi $\mathcal{H}$, non dai dati specifici. Una misura più raffinata è la **Rademacher Complexity**, che misura quanto bene le ipotesi in $\mathcal{H}$ possono correlare con rumore casuale sui dati di training stessi — adattandosi alla distribuzione effettiva dei dati anziché al worst case. La complessità di Rademacher empirica di $\mathcal{H}$ su un dataset $S = \{x_1, \ldots, x_m\}$ è:

$$\hat{\mathcal{R}}_S(\mathcal{H}) = \mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{h\in\mathcal{H}} \frac{1}{m}\sum_{i=1}^{m} \sigma_i h(x_i)\right]$$

dove $\sigma_1, \ldots, \sigma_m$ sono variabili di **Rademacher** — variabili indipendenti uniformi in $\{-1, +1\}$ — che modellano il "rumore casuale". L'interpretazione è elegante: $\hat{\mathcal{R}}_S(\mathcal{H})$ misura in media quanto bene la classe più ricca di $\mathcal{H}$ possa adattarsi a etichette completamente casuali sui dati di training. Se $\mathcal{H}$ contiene ipotesi molto espressive, il supremo sarà vicino a 1 perché esiste sempre un'ipotesi che correla con qualunque pattern casuale; se $\mathcal{H}$ è ristretta, il supremo sarà vicino a zero perché nessuna ipotesi riesce a fittare etichette casuali. La complessità di Rademacher $\mathcal{R}_m(\mathcal{H}) = \mathbb{E}_S[\hat{\mathcal{R}}_S(\mathcal{H})]$ (dove l'aspettativa è sulla scelta casuale del dataset) lega il gap di generalizzazione: con probabilità almeno $1-\delta$, per ogni $h \in \mathcal{H}$, $L_{\mathcal{D}}(h) \leq \hat{L}_S(h) + 2\mathcal{R}_m(\mathcal{H}) + O(\sqrt{\log(1/\delta)/m})$.

355.2 · Il Lemma di Contrazione e la Rademacher per le Reti Neurali

La proprietà di contrazione di Ledoux-Talagrand è la chiave per calcolare la complessità di Rademacher di composizioni di funzioni: se ϕ è una funzione Lipschitz con costante L, allora $\mathcal{R}_m(\phi \circ \mathcal{F}) \leq L \cdot \mathcal{R}_m(\mathcal{F})$. Applicato in cascata a una rete neurale layer per layer, questo lemma mostra che la complessità di Rademacher di una rete d-layer con pesi normati da una costante B scala come $O(B^d/\sqrt{m})$ — un bound che dipende dalla norma dei pesi, non dal numero di parametri. Questa osservazione è alla base dei bound di generalizzazione *norm-based* per le reti neurali: reti con pesi di piccola norma generalizzano bene anche se hanno molti parametri, perché la loro capacità effettiva (misurata dalla Rademacher Complexity) è limitata dalla norma dei pesi, non dalla dimensionalità.

PAC-Bayes: Generalizzazione per Distribuzioni di Ipotesi

Bound di generalizzazione per classificatori stocastici — la divergenza KL come penalità di complessità, il teorema di McAllester e le applicazioni alle reti neurali.

356.1 · Il Framework PAC-Bayes di McAllester

I bound di generalizzazione classici — VC theory, Rademacher — si applicano a singole ipotesi deterministiche. Il framework **PAC-Bayes** (McAllester 1998, 1999) generalizza questi bound

alle distribuzioni di ipotesi — i classificatori stocastici che campionano un'ipotesi dalla distribuzione posteriore prima di predire. Questo framework è naturalmente adatto all'inferenza bayesiana ma anche alle reti neurali con tecniche di regolarizzazione come il dropout, che in alcuni regimi è equivalente a un classificatore stocastico.

Il teorema fondamentale di McAllester afferma che per qualunque distribuzione prior P su $\mathcal{H}$ fissata prima di vedere i dati, con probabilità almeno $1-\delta$ su scelta del dataset S di dimensione m, per qualunque distribuzione posterior Q su $\mathcal{H}$:

$$L_{\mathcal{D}}(Q) \leq L_S(Q) + \sqrt{\frac{\mathrm{KL}(Q\|P) + \ln(m/\delta)}{2(m-1)}}$$

dove $L_S(Q) = \mathbb{E}_{h\sim Q}[\hat{L}_S(h)]$ è l'errore empirico atteso della posterior e $L_{\mathcal{D}}(Q) = \mathbb{E}_{h\sim Q}[L_{\mathcal{D}}(h)]$ è l'errore vero atteso. Il termine $\mathrm{KL}(Q\|P)$ funge da penalità di complessità: quanto più la posterior si discosta dalla prior, tanto più è necessario un dataset grande per garantire la generalizzazione. La bellezza del bound PAC-Bayes rispetto ai bound classici è che è *calcolabile numericamente*: dato un dataset, una prior gaussiana, e una rete neurale addestrata, si può calcolare esplicitamente il valore numerico del bound, ottenendo una garanzia di generalizzazione quantitativa e non solo asintotica.

356.2 · PAC-Bayes per le Reti Neurali: il Bound di Dziugaite e Roy

Dziugaite e Roy (2017) hanno applicato il framework PAC-Bayes alle reti neurali in modo diretto e quantitativo: si sceglie una prior gaussiana isotropa $P = \mathcal{N}(\mathbf{0}, \lambda^2 I)$ sulle reti, si addestra una rete con tecniche che minimizzano sia l'errore empirico sia la KL dalla prior — di fatto un training con regolarizzazione L2 avanzato — e si calcola il bound numerico. I risultati mostrano bound di generalizzazione non banali (inferiori al 20-30%) su MNIST, confermando che le garanzie PAC-Bayes possono essere quantitativamente significative per le reti neurali reali. Più recentemente, il **metodo di ottimizzazione del bound PAC-Bayes** (Thiemann et al. 2017) minimizza direttamente il bound come obiettivo di training — invece di minimizzare solo l'errore empirico — producendo classificatori con garanzie di generalizzazione più strette a spese di una leggera perdita in accuratezza empirica.

Compression Bounds: la Generalizzazione come Compressione

Il teorema di Occam, i bound di compressione di Littlestone-Warmuth e la connessione con MDL e teoria dell'informazione.

357.1 · L'Algoritmo di Occam e i Bound di Compressione

Un principio profondo lega la capacità di comprimere i dati alla capacità di generalizzare. Il **Rasoio di Occam** — nella sua versione quantitativa — afferma che ipotesi più semplici (che possono essere descritte con meno bit) generalizzano meglio. Il **Lemma di Occam** (Blumer et al. 1987) ne è la formalizzazione: se esiste un algoritmo che, dato un dataset di m esempi, produce un'ipotesi h che azzerano l'errore empirico e che può essere descritta con b bit, allora con probabilità almeno $1-\delta$, $L_{\mathcal{D}}(h) \leq O(\sqrt{(b+\log(1/\delta))/m})$. La dimensione della descrizione in bit prende il posto della VC dimension come misura di complessità — e per ipotesi con struttura semplice (alberi di decisione corti, reti sparse) può essere molto più piccola della VC dimension.

I **bound di compressione** (Littlestone e Warmuth 1986, Floyd e Warmuth 1995) generalizzano il Lemma di Occam: se l'algoritmo di apprendimento può comprimere il dataset di m esempi in $k < m$ esempi "*campioni di compressione*" che determinano univocamente l'ipotesi prodotta, allora l'errore di generalizzazione è al più $O(k \log m/m)$. Questo bound è interessante perché dipende dal numero di esempi di supporto k, non dalla dimensione della classe di ipotesi: per le SVM, i campioni di compressione sono i support vectors, e il bound di compressione coincide con il leave-one-out bound $O(|SV|/m)$ discusso nella §330. Per le reti neurali, i campioni di compressione corrispondono a un sottoinsieme degli esempi di training che determina i pesi ottimali — un'interpretazione che connette la generalizzazione neurale con la teoria dell'informazione e con il principio di compressione minima della lunghezza descrittiva (MDL).

357.2 · Minimum Description Length e l'Identificazione del Modello

Il principio **Minimum Description Length (MDL)** (Rissanen 1978, 1986) formalizza l'inferenza statistica come problema di compressione: il miglior modello per un dataset è quello che permette di codificarlo con il minor numero di bit. La lunghezza di descrizione di un dataset D sotto un modello M è $L(D, M) = L(M) + L(D|M)$ dove $L(M)$ è il costo di codifica del modello stesso (la sua complessità) e $L(D|M)$ è il costo di codifica dei dati dato il modello (essenzialmente $-\log P(D|M)$). Il MDL seleziona il modello $\hat{M} = rg\min_M L(D, M)$ — una forma di regolarizzazione implicita che penalizza la complessità del modello. La versione raffinata **Normalized Maximum Likelihood (NML)** sceglie la distribuzione universale $p_{mNML}(D) =$

$P(D|\hat{heta}(D))/\sum_{D'} P(D'|\hat{heta}(D'))$ dove la normalizzazione avviene su tutti i possibili dataset della stessa dimensione — producendo la distribuzione che minimizza il regret universale di codifica. Il termine di normalizzazione $\log \sum_{D'} P(D'|\hat{\theta}(D'))$ è chiamato **complessità parametrica** o **COMP**, e coincide asintoticamente con $\frac{k}{2}\log(n/2\pi) + \log|\det I(\hat{\theta})|^{1/2}$ (dove k è il numero di parametri e $I(\hat{\theta})$ è la matrice di Fisher valutata al MLE) — essenzialmente il criterio BIC di Schwarz con l'aggiunta del termine di Fisher che misura la forma della likelihood. MDL e BIC condividono la penalizzazione $\frac{k}{2}\log n$ del numero di parametri, ma MDL è preferibile perché derivato da principi di codifica senza assumere prior bayesiane, ed è applicabile a modelli non parametrici dove il numero di parametri non è ben definito.

Implicit Regularization nel Deep Learning

Perché le reti sovra-parametrizzate generalizzano — bias implicito di SGD, la norma minima e il fenomeno della doppia discesa.

358.1 · Il Paradosso della Sovra-Parametrizzazione

Le reti neurali moderne hanno miliardi di parametri e vengono addestrate su dataset di milioni di esempi — un regime in cui i parametri superano di gran lunga i campioni, che i teoremi classici della statistica identificano come zona di inevitabile *sovra-adattamento*. Eppure empiricamente queste reti generalizzano notevolmente bene. Questa osservazione apparentemente contraddittoria — che rimase senza spiegazione teorica per anni — è ora meglio compresa attraverso il concetto di **regolarizzazione implicita**: l'algoritmo di ottimizzazione SGD, anche in assenza di regolarizzazione esplicita, tende a trovare non un generico minimizzatore dell'errore empirico ma il minimizzatore con proprietà strutturali specifiche che lo rendono ben generalizzante.

Il risultato fondamentale per i problemi lineari è il seguente: su un sistema sovra-determinato con infinite soluzioni a errore empirico zero, il gradient descent con inizializzazione vicina all'origine converge alla soluzione a **norma ℓ^2 minima** — equivalente alla pseudo-inversa di Moore-Penrose. La norma minima è una forma di regolarizzazione implicita che favorisce soluzioni "semplici" anche senza penalità esplicita. Zhang et al. (2017) hanno dimostrato sperimentalmente che le reti neurali possono memorizzare qualunque dataset di training con etichette casuali, ma quando addestrate su dati con struttura reale tendono verso soluzioni semplici — la norma dei pesi rimane piccola nonostante la sovra-parametrizzazione. Il fenomeno della **doppia discesa** (Belkin et al. 2019, Nakkiran et al. 2020) mostra che la curva di errore in funzione della capacità del modello ha due discese: la prima segue il classico tradeoff bias-

varianza (l'errore aumenta nella zona di interpolazione), la seconda avviene oltre il punto di interpolazione dove il modello ha abbastanza capacità da memorizzare il training set — nella zona sovra-parametrizzata l'errore scende di nuovo perché il modello trova soluzioni a bassa norma che generalizzano bene.

358.2 · Neural Collapse: la Geometria dell'Ultimo Layer

Un fenomeno di regolarizzazione implicita particolarmente sorprendente è la **Neural Collapse** (Papyan, Han e Donoho 2020): quando una rete di classificazione viene addestrata fino alla convergenza completa su dati bilanciati — ben oltre il punto di zero training error — le rappresentazioni dell'ultimo layer collassano verso una struttura geometrica molto precisa. Le K classi di un problema di classificazione a K classi vengono rappresentate da K vettori di classe medi che formano un *simplex equiangolare di norma massima* (ETF, Equiangular Tight Frame): tutti i vettori hanno la stessa norma, e i prodotti interni tra vettori diversi sono uguali e negativi $\langle \mathbf{h}_i, \mathbf{h}_j angle = -1/(K-1)$ per $ieqj$. Nello stesso tempo, la matrice dei pesi dell'ultimo layer lineare converge a una matrice che assegna a ogni classe il proprio vettore di classe medio (dualità self-duality). Questa struttura geometrica è la stessa che emerge dalla soluzione analitica del problema di classificazione con perdita di cross-entropy e regolarizzazione L2 nel limite di convergenza completa — la regolarizzazione implicita di SGD guida spontaneamente le rappresentazioni verso la struttura ottimale senza che questa sia mai imposta esplicitamente. La Neural Collapse ha implicazioni pratiche per il transfer learning: le rappresentazioni dell'ETF sono ottimali per il few-shot learning perché massimizzano la separazione angolare tra classi, e tecniche di training che inducono esplicitamente la Neural Collapse hanno migliorato le prestazioni few-shot e long-tail su benchmark di riconoscimento visivo.

Algoritmi Online con Feedback Ritardato e Parziale

Il regime più generale dell'online learning — banditi con ritardo, feedback parziale strutturato e il modello degli esperti con informazione limitata.

385.1 · Il Modello del Feedback Ritardato

I framework OCO e bandit classici assumono che il feedback sulla perdita sia disponibile immediatamente dopo ogni azione. In molti scenari reali questo non avviene: nelle aste pubblicitarie online il click-through rate viene osservato con ritardo rispetto alla visualizzazione dell'annuncio; nei sistemi di raccomandazione il feedback dell'utente arriva ore o giorni dopo;

nei sistemi di trading ad alta frequenza la conferma di esecuzione può arrivare con un ritardo di millisecondi ma nel contesto dell'algoritmo è un'intera iterazione. Il modello del **feedback ritardato** (Joulani, Gyorgy e Szepesvari 2013) formalizza questo scenario: ad ogni round t si gioca un'azione a_t, ma la perdita $\ell_t(a_t)$ viene rivelata solo al round $t+d_t$ dove $d_t \geq 0$ è il ritardo, potenzialmente variabile e non noto in anticipo. Il bound sul regret si degrada di un fattore che dipende dalla distribuzione dei ritardi: per ritardi uniformemente bounded da d, il regret di Hedge con ritardo è $O(\sqrt{T(d+1)\log N})$ — un fattore $\sqrt{d+1}$ rispetto al caso senza ritardo. Questo risultato è quasi ottimale: esiste un lower bound $\Omega(\sqrt{Td\log N/\log d})$ che mostra che il ritardo peggiora il regret in modo sottolineare nel ritardo stesso, non lineare — una buona notizia per le applicazioni dove il ritardo è moderato rispetto all'orizzonte temporale T.

385.2 · Banditi con Feedback di Grafo

Un'ulteriore generalizzazione del modello bandit riguarda la struttura del feedback: nei banditi standard, scegliendo l'azione i si osserva solo la perdita di i; nei banditi con feedback di grafo (Mannor e Shamir 2011) si osserva la perdita di tutti i vicini di i in un grafo di feedback $G = (V, E)$. Scegliere un nodo con molti vicini fornisce più informazioni. Il regret dipende dal numero di **dominanting sets** del grafo: un insieme indipendente $\alpha(G)$ (massimo insieme di nodi senza archi in comune) e il numero di dominanza $\gamma(G)$ (minimo insieme che copre tutti i nodi). Per grafi densi (molti archi), il regret è vicino a quello del setting con piena osservabilità $O(\sqrt{T\log N})$; per grafi sparsi (pochi archi), si avvicina al regret bandit $O(\sqrt{KT})$. Questo framework modella naturalmente le reti di distribuzione dove scegliere di produrre un certo prodotto rivela informazioni sulle preferenze per prodotti simili, o le reti sociali dove il word-of-mouth propaga l'informazione sulle preferenze degli utenti.

Portfolio Selection e la Connessione con l'Online Learning

Il problema del gestore di portafoglio come istanza OCO — il criterio di Kelly, l'algoritmo Universal Portfolio di Cover e il regret logaritmico.

386.1 · Universal Portfolio: Apprendimento Online senza Assunzioni sul Mercato

Thomas Cover (1991) ha formulato il problema della **gestione di portafoglio online** come un problema di apprendimento competitivo: un investitore alloca il proprio capitale su K asset ad ogni round senza conoscere i prezzi futuri, e vuole fare bene rispetto al miglior portafoglio fisso in hindsight — la strategia che avrebbe fatto meglio sapendo già tutti i prezzi. La performance

è misurata dal rendimento logaritmico cumulativo (il criterio di Kelly) anziché dalla somma delle perdite, perché la crescita del capitale è geometrica, non aritmetica. L'algoritmo **Universal Portfolio** di Cover mantiene una distribuzione di portafogli campionati uniformemente dal simplesso delle allocazioni, pesa ogni portafoglio per la sua performance storica, e investe la media pesata. Il risultato notevole è che il rimpianto del rendimento logaritmico è $O(\log T)$ — logaritmico in T invece che $\sqrt{T}$, possibile perché il problema ha struttura concava forte (il rendimento logaritmico è concavo nelle allocazioni) e l'esperto ottimale varia lentamente. Questo bound logaritmico è ottimale: non è possibile fare meglio di $\Omega(\log T)$ in questo setting. La connessione con la Rademacher complexity (§355) e il PAC-Bayes (§356) è diretta: il Universal Portfolio è essenzialmente l'algoritmo Bayesiano con prior uniforme sul simplesso, e il suo bound di regret è il bound PAC-Bayes applicato alla distribuzione dei portafogli. Algoritmi più moderni come EG+ e ONS (Online Newton Step) ottengono lo stesso bound logaritmico con costo computazionale inferiore grazie alla struttura della funzione di guadagno logaritmico.

Multi-Armed Bandits: Thompson Sampling, Bayes-UCB e Banditi Bayesiani

L'approccio bayesiano ai banditi stocastici — la posterior come strumento di esplorazione, la teoria del regret bayesiano e le applicazioni ai test A/B.

387.1 · Thompson Sampling: Esplorazione come Campionamento Posteriore

Il **Thompson Sampling** (Thompson 1933, riscoperto da Chapelle e Li 2011) è uno degli algoritmi bandit più antichi e al tempo stesso più moderni: si mantiene una distribuzione bayesiana a posteriori sui parametri di reward di ogni azione, e ad ogni round si campiona un set di parametri dalla posteriore e si sceglie l'azione con il reward stimato più alto sotto quei parametri. L'intuizione è elegante: si agisce *come se* i parametri campionati fossero quelli veri — una forma di "ottimismo campionato" che si traduce in esplorazione automatica quando l'incertezza è alta (la posteriore è larga, diversi campioni danno rankings diversi) e in sfruttamento quando l'incertezza è bassa (la posteriore è concentrata, tutti i campioni concordano). Per il problema di bandit Bernoulli con prior Beta(α_k, β_k) per ogni azione, il Thompson Sampling si aggiorna in modo bayesiano con aggiornamento coniugato: dopo una reward 1 sull'azione k, si aggiorna $\alpha_k \leftarrow \alpha_k + 1$; dopo una reward 0, $\beta_k \leftarrow \beta_k + 1$. Il regret atteso di Thompson Sampling è $O(\sum_k \log T / \Delta_k)$ dove $\Delta_k = \mu^* - \mu_k$ è il gap tra l'azione ottimale e l'azione k — lo stesso bound di UCB1, e anch'esso ottimale rispetto al lower bound di Lai-Robbins (§339). Empiricamente, Thompson Sampling tende a superare UCB1 grazie a

un minor tasso di esplorazione nella pratica: il campionamento dalla posteriore fornisce esplorazione solo nella misura in cui l'incertezza è effettivamente rilevante, mentre UCB esplora in modo più conservativo deterministo. Per questo Thompson Sampling è l'algoritmo di riferimento per i test A/B online e i sistemi di raccomandazione, dove l'efficienza dell'esplorazione si traduce direttamente in revenue durante la fase di apprendimento.

387.2 · Bayes-UCB e la Connessione tra Probabilità e Bound di Confidenza

Il **Bayes-UCB** (Kaufmann, Cappé e Garivier 2012) unifica l'approccio frequentista di UCB e quello bayesiano di Thompson Sampling: invece di campionare dalla posteriore, si usa un quantile alto della posteriore come upper confidence bound. Formalmente, al round t si sceglie $a_t = \arg\max_k Q(1 - 1/(t \log T), \theta_k | \mathcal{D}_{t-1})$ dove $Q(\alpha, \theta|\mathcal{D})$ è il quantile α della distribuzione posteriore su θ_k. Per quantili vicini a 1 (alta confidenza), questo bound è quasi identico al sample massimo della posteriore — simile a Thompson Sampling. Per quantili molto vicini a 1 (estrema ottimismo), si avvicina a UCB. Bayes-UCB ottiene il bound di Lai-Robbins con costanti migliori di UCB1 su molte famiglie di distribuzioni, ed è facilmente applicabile a qualunque modello bayesiano per cui i quantili posteriori siano calcolabili — il che lo rende il punto di partenza naturale per bandit con struttura complessa (banditi lineare, banditi con correlazioni tra azioni, banditi su grafi).

Mixture of Experts Classico: Gating Networks e la Specializzazione

Il principio del "dividi e governa" applicato alle reti neurali — gate differenziabili, EM per MoE e il legame con il clustering morbido.

359.1 · L'Idea Fondamentale del Divide-and-Conquer Neurale

Il **Mixture of Experts (MoE)**, introdotto da Jacobs, Jordan, Nowlan e Hinton (1991), nasce dall'osservazione che problemi di apprendimento complessi possono essere decomponibili in sottoproblemi più semplici, ciascuno risolvibile da un "esperto" specializzato. Invece di addestrare un'unica rete grande a gestire l'intero dominio di input, si addestrano K reti "esperte" $\{E_k(\mathbf{x}; \theta_k)\}_{k=1}^K$ insieme a una "rete di gating" $G(\mathbf{x}; \phi)$ che produce una distribuzione di probabilità su quali esperti sono rilevanti per ogni input. L'output finale è la media pesata degli output degli esperti, con pesi forniti dalla rete di gating:

$$y_{\text{MoE}}(\mathbf{x}) = \sum_{k=1}^{K} G_k(\mathbf{x};\phi)\, E_k(\mathbf{x};\theta_k), \qquad G_k \geq 0,\ \sum_k G_k = 1$$

Il gating morbido con $G_k > 0$ per tutti gli esperti è il regime del MoE classico: ogni input riceve contributi da tutti gli esperti, pesati dalla loro rilevanza. Questo porta a un'ottimizzazione difficile con soluzioni degeneri — il gating tende a ignorare la maggior parte degli esperti, convergendo verso un singolo esperto dominante. L'algoritmo EM fornisce l'alternativa naturale: nel passo E si calcolano i pesi di responsabilità $r_{ik} = G_k(\mathbf{x}_i;\phi)\, P(y_i \mid \mathbf{x}_i;\theta_k) / \sum_j G_j P(y_i \mid \mathbf{x}_i;\theta_j)$ che indicano quanto l'esperto k è responsabile dell'esempio i; nel passo M si aggiornano i parametri degli esperti con regressione pesata e i parametri del gating tramite massimizzazione della likelihood. La connessione con il clustering morbido (Gaussian Mixture Models) è diretta: se gli esperti sono gaussiane e il gating è la probabilità di appartenenza al cluster, l'EM per MoE coincide con l'EM per GMM.

Sparse MoE e Top-K Routing nei Transformer

L'attivazione sparsa come chiave della scalabilità — il gating hard, il problema del carico bilanciato e la derivabilità attraverso operazioni discrete.

360.1 · Il Salto dalla Miscela Morbida al Routing Duro

L'integrazione del Mixture of Experts nell'architettura Transformer — per costruire modelli di linguaggio con centinaia di miliardi di parametri a un costo computazionale sostenibile — richiede una modifica fondamentale al gating classico: invece di attivare tutti gli esperti per ogni token (computazionalmente equivalente a avere una rete K volte più grande), si attivano solo i **top-k** esperti per ogni token, tipicamente $k = 1$ o $k = 2$. In questo regime *sparse*, il costo computazionale per token è $O(k/K)$ del costo del dense MoE, permettendo di aumentare il numero di esperti (e quindi la capacità totale del modello) senza aumentare il costo per token. La condizione di sparsità introduce però una difficoltà tecnica cruciale: la selezione top-k è un'operazione discreta non differenziabile rispetto ai logit del gating. La soluzione standard è l'aggiunta di rumore gaussiano prima del top-k: $G(\mathbf{x}) = \text{softmax}(\text{topk}(h(\mathbf{x}) + \epsilon \cdot \mathcal{N}(\mathbf{0}, \mathbf{1})))$ dove $h(\mathbf{x}) = \mathbf{x}W_g$ sono i logit di gating e il rumore permette la propagazione approssimata del gradiente attraverso la selezione discreta tramite il trucco dello stimatore straight-through.

L'ostacolo principale del top-k routing sparso è la tendenza al **collasso degli esperti**: il gating tende a instradare quasi tutti i token verso un piccolo sottoinsieme di esperti, mentre gli altri rimangono sotto-utilizzati e non apprendono nulla. Il meccanismo è un circolo vizioso: gli esperti che ricevono più token apprendono più velocemente, diventano più competenti, vengono quindi preferiti dal gating, ricevono ancora più token, e così via. Questo problema è analogo alla "vincitore porta tutto" del softmax — la scelta del top-k amplifica le piccole differenze nei logit del gating. La soluzione introdotta da Shazeer et al. (2017) è una **loss di bilanciamento del carico** aggiuntiva che penalizza la concentrazione del routing su pochi esperti. La loss più comune calcola la correlazione tra la frequenza di selezione di ogni esperto e la probabilità media di selezione stimata dal softmax: $\mathcal{L}_{\text{aux}} = K \sum_{k=1}^{K} f_k \cdot P_k$ dove f_k è la frazione di token instradati all'esperto k e $P_k = \frac{1}{T} \sum_t G_{tk}$ è la probabilità media di gating. Minimizzare questa loss incentiva la distribuzione uniforme del carico mantenendo la differenziabilità fine al gating — una delle principali innovazioni ingegneristiche che ha reso praticamente stabile il training degli Sparse MoE su scala industriale.

Switch Transformer, GLaM e Mixtral

I grandi modelli MoE in pratica — $k = 1$, capacità adattiva, esecuzione su hardware distribuito e i risultati empirici.

361.1 · Switch Transformer: $k = 1$ e la Semplicità Vincente

Il **Switch Transformer** (Fedus, Zoph e Dean, Google Brain 2021) ha dimostrato che la scelta $k = 1$ — instradare ogni token a un singolo esperto invece dei due standard — non solo è sufficiente per ottenere ottimi risultati ma semplifica il training e riduce la varianza dei gradienti. Il modello sceglie l'esperto con il logit di gating più alto per ogni token e usa una loss di bilanciamento semplificata. Il vantaggio principale di $k = 1$ è la fattibilità su hardware distribuito: con due esperti per token, ogni token deve essere instradato a due schede hardware diverse, richiedendo comunicazione punto-a-punto costosa; con un singolo esperto, ogni token rimane su un singolo dispositivo. Switch Transformer con 1.6 trilioni di parametri totali (ma costo per token equivalente a un modello da 2 miliardi di parametri densi) ha dimostrato un'accelerazione nel pre-training del 7 volte rispetto al modello T5 base, stabilendo che il MoE sparso scala eccellentemente sia in qualità che in efficienza. La chiave del successo rispetto ai tentativi precedenti di MoE a grande scala è la semplicità: meno iperarametri, meno trucchi di

stabilizzazione, e un'architettura che si adatta naturalmente alla parallelizzazione su centinaia di acceleratori.

361.2 · Mixtral e il Renaissance del MoE Open-Source

Nel 2023, Mistral AI ha rilasciato **Mixtral 8**$\times 7B$, $un modello MoE sparso con 8 esperti per layer e \mathbf{k=2}$$--$$-in ogni layer si attivano 2 esperti su 8 per ogni token. Con 46.7 miliardi di parametri totali ma solo 12.9 miliardi a$ 3.5e $Llama 270B su molti benchmark standard. Mixtral ha dimostrato empiricamente un fatto importante :$ $il MoE sparso non riservato ai giganti industriali con centinaia di acceleratori ma praticabile anche a scale mode$ $source competitivi a costo computazionale contenuto. La sua architettura essenzialmente una Mistral 7B dove$ $8 FFN indipendenti, con ogni FFN avente le stesse dimensioni della FFN originale$$---un'impostazione mi$

Load Balancing Avanzato e Esperti Gerarchici

Oltre la loss ausiliaria — soft MoE, expert choice routing e la connessione con l'attenzione sparsa.

362.1 · Expert Choice Routing: Capovolgere la Prospettiva

Il routing standard nei Sparse MoE è *token-choice*: ogni token sceglie i suoi top-k esperti. Un approccio alternativo, proposto da Zhou et al. (2022) con il nome di Expert Choice Routing, capovolge la prospettiva: invece di avere ogni token che sceglie gli esperti, è ogni esperto che sceglie i suoi top-k token preferiti dal batch. Questo capovolgimento elimina automaticamente il problema del carico sbilanciato — ogni esperto riceve esattamente k token — senza richiedere la loss ausiliaria. Il meccanismo permette una distribuzione adattiva dell'attenzione computazionale: i token difficili o informativi tendono ad essere selezionati da più esperti (poiché hanno logit di gating alti per molti esperti), ricevendo più elaborazione, mentre i token semplici possono essere selezionati da pochi esperti. Questa proprietà è desiderabile perché rispecchia l'intuizione che diversi token richiedono diversi livelli di elaborazione — un'idea che connette il MoE con i meccanismi di adaptive computation time. Il vantaggio si manifesta concretamente: Expert Choice ottiene la stessa qualità del routing standard con il 20-30% di FLOPS in meno, perché i token difficili ricevono più elaborazione mentre i token facili ne ricevono meno, distribuendo il budget computazionale in modo adattivo.

362.2 · Soft MoE: Combinazione Differenziabile senza Routing Discreto

Una soluzione radicalmente diversa al problema del gradiente discontinuo è il Soft MoE (Puigcerver et al. 2023, Google DeepMind): eliminare completamente la selezione discreta e sostituirla con una combinazione morbida pienamente differenziabile. Nel Soft MoE, ogni "slot" di un esperto non riceve un singolo token ma una combinazione convessa di tutti i token del batch, con pesi determinati da un meccanismo di attenzione: $\tilde{\mathbf{x}}_s = \sum_t \phi_{ts} \mathbf{x}_t$ dove $\Phi = \text{softmax}(\mathbf{X} W_\phi / \sqrt{d})$ è la matrice dei pesi di mescolamento e W_ϕ è una matrice apprendibile con una colonna per ogni slot degli esperti. L'output del layer viene poi ricomposto rimescolando gli output degli slot con la matrice di dispatch trasposta: $\tilde{\mathbf{y}}_t = \sum_{k,s} \phi_{t,ks} E_k(\tilde{\mathbf{x}}_{ks})$. Poiché l'intera operazione è differenziabile — nessuna selezione discreta, nessun gradiente approssimato con straight-through — il Soft MoE elimina la necessità della loss ausiliaria di bilanciamento: la distribuzione del carico emerge naturalmente dall'ottimizzazione. Il prezzo è che tutti gli esperti vengono calcolati per ogni batch, perdendo il risparmio computazionale del routing sparso; Soft MoE è quindi più adatto al training che all'inferenza, e architetture ibride che usano Soft MoE in training e routing sparso in inferenza sono un'area di sviluppo attiva.

MoE per la Visione, il Multimodale e le Scienze

Mixture of Experts oltre i modelli linguistici — V-MoE, Unified-IO e la specializzazione per dominio.

363.1 · Vision MoE e la Specializzazione per Patch

Il successo del Sparse MoE nei modelli linguistici ha naturalmente ispirato la sua applicazione ai Vision Transformer. Il V-MoE (Riquelme et al. 2021) sostituisce i layer FFN densi dei ViT con Sparse MoE con $k = 2$ e mostra che i modelli vision traggono ancora più vantaggio dal MoE rispetto ai modelli linguistici: gli esperti si specializzano spontaneamente per diversi tipi di contenuto visivo — alcuni esperti si attivano prevalentemente per patch contenenti volti, altri per texture naturali, altri per bordi e strutture geometriche. Questa specializzazione non è imposta ma emerge dall'ottimizzazione dell'obiettivo di classificazione, a conferma che la struttura gerarchica e composizionale delle scene visive si presta naturalmente alla decomposizione in esperti specializzati. I modelli Unified-IO e Gato di DeepMind

hanno esteso questa idea al setting multi-task e multimodale: un singolo modello MoE gestisce testo, immagini, audio e sequenze di azioni, con gli esperti che si specializzano sia per modalità che per tipo di task. La sfida principale in questo contesto è che la distribuzione dei token varia enormemente tra modalità diverse, rendendo il load balancing ancora più critico: un batch che mescola testo e immagini ha token con distribuzioni di attivazione molto diverse, e senza load balancing gli esperti tendono a specializzarsi per modalità in modo così estremo da perdere la capacità di generalizzazione cross-modale.

363.2 · Scaling Laws per i Modelli MoE

Una delle domande più importanti sul MoE sparso è come si scalano le sue prestazioni con il numero di esperti, la dimensione dei singoli esperti, e il volume di dati di training. L'analisi delle leggi di scala per i MoE (Clark et al. 2022) ha prodotto risultati non ovvi: aggiungere esperti migliora la qualità ma con rendimenti decrescenti — il beneficio del raddoppio del numero di esperti equivale approssimativamente a un incremento del 10-20% nel numero di parametri di un modello denso equivalente. Questo implica che i MoE sono più efficienti dei modelli densi nel regime di training a budget fisso (dove si confrontano modelli con lo stesso FLOP budget di addestramento), ma meno efficienti in termini di parametri totali rispetto alla qualità raggiunta. La regola empirica è che un MoE con K esperti di dimensione d si comporta come un modello denso di dimensione $\approx d \log K$ — la capacità aggiuntiva degli esperti contribuisce logaritmicamente, non linearmente. Questo effetto di rendimento decrescente è spiegato teoricamente dall'entropia del routing: con K esperti e top-k routing, ogni token esplora in media solo k combinazioni su $\binom{K}{k}$ possibili — la specializzazione degli esperti è limitata dalla rara attivazione di combinazioni di esperti diverse. Metodi che aumentano artificialmente la diversità delle combinazioni — come Expert Dropout (disabilitare casualmente alcuni esperti durante il training) o la randomizzazione del routing durante i primi passi — hanno mostrato miglioramenti consistenti sulla qualità finale dei modelli MoE a larga scala.

Random Fourier Features e Approssimazione di Nyström

Due strade per scalare i kernel methods — il teorema di Bochner per le RFF e l'approssimazione a basso rango con punti landmark per Nyström.

364.1 · Random Fourier Features: il Teorema di Bochner in Azione

La §329 ha introdotto le Random Fourier Features come tecnica per scalare il kernel SVM al caso con n grande. Vale la pena sviluppare la teoria in maggiore profondità perché le RFF hanno implicazioni molto più ampie: esse forniscono un ponte tra i kernel methods e i modelli lineari, permettendo di "kernelizzare" qualunque algoritmo lineare a un costo computazionale $O(nD)$ invece di $O(n^2)$, dove D è il numero di feature casuali. Il Teorema di Bochner afferma che $k : \mathbb{R}^d \to \mathbb{R}$ è un kernel positivo semi-definito stazionario (cioè $k(\mathbf{x}, \mathbf{y}) = k(\mathbf{x} - \mathbf{y})$) se e solo se è la trasformata di Fourier di una misura di probabilità non negativa $p(\boldsymbol{\omega})$: $k(\boldsymbol{\delta}) = \int_{\mathbb{R}^d} p(\boldsymbol{\omega}) e^{i\boldsymbol{\omega}^T \boldsymbol{\delta}}\, d\boldsymbol{\omega}$. Prendendo la parte reale (che è legittimo per kernel simmetrici reali) e scrivendo $e^{i\boldsymbol{\omega}^T\boldsymbol{\delta}} = \cos(\boldsymbol{\omega}^T\mathbf{x} - \boldsymbol{\omega}^T\mathbf{y}) \cdot$ (termine reale), si ottiene la decomposizione: $k(\mathbf{x}, \mathbf{y}) = \mathbb{E}_{\boldsymbol{\omega}\sim p}[2\cos(\boldsymbol{\omega}^T\mathbf{x} + b)\cos(\boldsymbol{\omega}^T\mathbf{y} + b)]$ per $b \sim \mathcal{U}[0, 2\pi]$. Campionando D coppie $(\boldsymbol{\omega}_j, b_j)$ i.i.d. e costruendo la mappa $z(\mathbf{x}) = \sqrt{2/D}(\cos(\boldsymbol{\omega}_1^T\mathbf{x} + b_1), \ldots, \cos(\boldsymbol{\omega}_D^T\mathbf{x} + b_D))^T$, si ottiene uno stimatore non biasato del kernel: $\mathbb{E}[z(\mathbf{x})^T z(\mathbf{y})] = k(\mathbf{x}, \mathbf{y})$. La concentrazione dello stimatore è controllata dalla disuguaglianza di Hoeffding: con $D = O(d\log(1/\varepsilon)/\varepsilon^2)$ feature, l'errore massimo di approssimazione su qualunque input in una palla di raggio r è al più ε con alta probabilità, permettendo di sostituire il kernel esatto con la mappa RFF in qualunque algoritmo kernelizzato senza degradazione significativa delle prestazioni.

364.2 · L'Approssimazione di Nyström

L'approssimazione di Nyström (Williams e Seeger 2001, nata originariamente nel contesto dei metodi agli elementi finiti) offre un approccio diverso per approssimare la matrice di Gram $K \in \mathbb{R}^{n\times n}$: invece di approssimare ogni valore del kernel con una mappa esplicita, si costruisce un'approssimazione di basso rango usando un sottoinsieme di $m \ll n$ punti *landmark* $\{u_1, \ldots, u_m\} \subset \mathcal{X}$. L'idea è che la matrice di Gram può essere approssimata come $K \approx K_{nm}K_{mm}^{-1}K_{mn}$ dove K_{nm} è la matrice $n \times m$ dei kernel tra i punti di training e i landmark, e K_{mm} è la matrice $m \times m$ dei kernel tra i landmark. Questa decomposizione ha rango esattamente m e può essere calcolata in $O(nm^2)$ — lo stesso scaling degli Sparse GP FITC. La scelta dei punti landmark influisce molto sulla qualità dell'approssimazione: i metodi

più semplici usano campionamento uniforme, quelli più sofisticati usano k-means clustering per selezionare landmark rappresentativi della distribuzione dei dati, oppure campionamento di leveraging scores (Mahoney e Drineas 2009) che pesa i punti proporzionalmente alla loro importanza nell'approssimazione a basso rango. La connessione con gli Sparse GP è immediata: l'approssimazione di Nyström per la matrice di Gram del kernel è esattamente la fattorizzazione a basso rango Q_{nn} usata in FITC (§319), con i punti landmark che corrispondono agli inducing points.

Deep Kernel Learning

Reti neurali come trasformazioni dello spazio delle feature per i GP — apprendimento end-to-end di rappresentazioni e kernel ibridi.

365.1 · La Fusione tra Rappresentazione Neurale e Processo Gaussiano

I Processi Gaussiani (§316) hanno eccellenti proprietà di quantificazione dell'incertezza e ottimizzazione degli iperparametri, ma dipendono criticamente dalla scelta del kernel — che deve essere progettato manualmente per ogni dominio. Le reti neurali apprendono rappresentazioni potenti ma non forniscono naturalmente stime di incertezza calibrate. Il Deep Kernel Learning (DKL) (Wilson, Hu, Salakhutdinov e Xing 2016) fonde i due paradigmi: una rete neurale $\phi_\theta(\mathbf{x})$ apprende una rappresentazione dell'input, e un GP con kernel standard (RBF, Matérn) opera nello spazio delle rappresentazioni neurali: $k_{\text{DKL}}(\mathbf{x}, \mathbf{x}') = k_{\text{base}}(\phi_\theta(\mathbf{x}), \phi_\theta(\mathbf{x}'))$. Il training end-to-end ottimizza congiuntamente i pesi θ della rete e gli iperparametri del kernel base massimizzando la log-marginal likelihood del GP — l'obiettivo standard per la selezione degli iperparametri nei GP. Il risultato è un modello che combina la potenza espressiva delle reti neurali nella costruzione della feature map con le garanzie bayesiane dei GP nell'incertezza predittiva. In pratica, il DKL è particolarmente efficace per i dati tabulari dove le relazioni tra feature sono complesse e non catturabili da kernel stazionari semplici, e per l'ottimizzazione bayesiana dove la qualità della quantificazione dell'incertezza è cruciale per l'efficienza dell'acquisizione.

365.2 · Neural Processes: Meta-Apprendimento con Incertezza Bayesiana

Un'estensione ambiziosa del Deep Kernel Learning è il framework dei Neural Processes (NP) (Garnelo, Schwarz, Rosenbaum et al. 2018), che punta a combinare la

flessibilità delle reti neurali con la coerenza probabilistica dei Processi Gaussiani nel contesto del meta-apprendimento. Un NP apprende una distribuzione su funzioni, ma usa una rete neurale per inferire la distribuzione posteriore date le osservazioni di contesto, anziché la formula chiusa del GP. L'architettura ha tre componenti. Un *encoder* aggrega le osservazioni di contesto $\{(x_i, y_i)\}_{i \in C}$ in una rappresentazione latente tramite un aggregatore permutation-invariante (tipicamente la media o un meccanismo di attenzione). Un *encoder variazionale* inferisce la distribuzione di un vettore latente globale $z \sim q(z|\mathbf{r})$ che cattura l'incertezza sull'intera funzione. Un *decoder* predice $y_t \sim p(y|z, x_t)$ per i punti target. Il training massimizza un ELBO che ricompensa sia l'accuratezza che la coerenza della distribuzione posteriore. Il limite principale dei NP base è che il vettore z è globale e non cattura le covarianze locali tra punti vicini, portando a stime di incertezza meno calibrate rispetto ai GP esatti. Gli Attentive Neural Processes (Kim et al. 2019) e i Gaussian Neural Processes (Bruinsma et al. 2021) affrontano questa limitazione con meccanismi di attenzione cross che modellano le dipendenze locali, avvicinando la qualità delle predizioni ai GP mantenendo la scalabilità delle reti neurali su dataset di grandi dimensioni.

Kernel per Sequenze, Grafi e Strutture Discrete

Misure di similarità per oggetti non vettoriali — string kernels, tree kernels, Weisfeiler-Lehman e graph kernels per la chemioinformatica.

366.1 · String Kernels: Confrontare le Sequenze senza Allineamento

Per oggetti strutturati come stringhe di caratteri (sequenze DNA, testo, proteine) o alberi (strutture sintattiche, strutture molecolari), non è ovvio come definire una rappresentazione vettoriale che preservi la struttura dell'oggetto. I kernel per strutture discrete affrontano questo problema direttamente nello spazio degli oggetti, calcolando la similarità tra due oggetti come una misura di quanto struttura condividono. Lo spectrum kernel (Leslie et al. 2002) per stringhe conta le sottosequenze di lunghezza k (i k-meri) in comune tra due sequenze: $k(\mathbf{s}, \mathbf{t}) = \sum_{\mathbf{u}:\,|\mathbf{u}|=k} c(\mathbf{u}, \mathbf{s})\, c(\mathbf{u}, \mathbf{t})$ dove $c(\mathbf{u}, \mathbf{s})$ conta le occorrenze del pattern u nella stringa s. Per le sequenze proteiche, questo kernel cattura la similarità tra famiglie proteiche che condividono motivi strutturali comuni. Lo weighted degree kernel

(Schölkopf e Ratsch 2004) pesa le corrispondenze per la lunghezza del pattern, privilegiando le corrispondenze esatte lunghe rispetto a quelle corte — utile per il riconoscimento di sequenze regolatorie nel DNA. Il calcolo efficiente di questi kernel usa le suffix arrays e i suffix trees, strutture dati per la ricerca di pattern in tempo $O(n)$.

366.2 · Graph Kernels e il Weisfeiler-Lehman

Per i grafi (molecole, reti metaboliche, grafi di co-citazione), i graph kernels misurano la similarità strutturale tra grafi. Il random walk kernel (Gärtner et al. 2003) conta le passeggiate casuali in comune tra due grafi; il shortest path kernel confronta le distribuzioni delle lunghezze dei cammini minimi; il Weisfeiler-Lehman kernel (Shervashidze et al. 2011) è tra i più potenti ed efficienti. WL usa il test di isomorfismo di Weisfeiler-Lehman come base: si assegna un'etichetta iniziale a ogni nodo (tipicamente il grado o un attributo), poi si aggiorna iterativamente l'etichetta di ogni nodo aggregando le etichette dei vicini (tramite hash). Dopo h round di aggiornamento, si conta il numero di label identiche tra i due grafi — la similarità WL di profondità h. WL ha un calcolo lineare nel numero di archi e si è dimostrato molto efficace per la classificazione di molecole organiche — il kernel preferito prima dell'avvento dei Graph Neural Network. La connessione tra WL e GNN è profonda: le architetture GNN standard (Xu et al. 2019) sono al massimo potenti quanto il test WL nel distinguere strutture grafiche non isomorfe, e i GNN più potenti (come il Relational Pooling e gli Higher-Order GNN) corrispondono a versioni ad ordine superiore del test WL.

Sketching e Metodi Randomizzati per l'Algebra Lineare

Tecniche Monte Carlo per sistemi lineari di grandi dimensioni — Johnson-Lindenstrauss, CountSketch e applicazioni al machine learning.

367.1 · Il Lemma di Johnson-Lindenstrauss e la Proiezione Casuale

Una conseguenza sorprendente della geometria degli spazi ad alta dimensione è che qualunque insieme di n punti in $\mathbb{R}^d$ può essere proiettato in $\mathbb{R}^k$ con $k = O(\log n/\varepsilon^2)$ dimensioni preservando tutte le distanze a meno di un fattore $(1 \pm \varepsilon)$. Questo è il contenuto del Lemma di Johnson-Lindenstrauss (1984): la proiezione ca-

suale con una matrice $\Phi \in \mathbb{R}^{k\times d}$ con entrate gaussiane i.i.d. $\mathcal{N}(0,1/k)$ soddisfa $(1-\varepsilon)\|\mathbf{x}_i-\mathbf{x}_j\|^2 \le \|\Phi\mathbf{x}_i-\Phi\mathbf{x}_j\|^2 \le (1+\varepsilon)\|\mathbf{x}_i-\mathbf{x}_j\|^2$ con probabilità almeno $1-2e^{-(\varepsilon^2-\varepsilon^3)k/4}$ — indipendentemente da d. La chiave è che la dimensione target k dipende solo dal numero di punti n e dalla precisione desiderata ε, non dalla dimensione originale d: anche per $d = 10^6$, è sufficiente proiettare a $k = O(\log n)$ dimensioni per preservare la struttura geometrica dell'insieme di punti. Il JL lemma è alla base di numerose tecniche di riduzione della dimensionalità per il machine learning: Locality Sensitive Hashing (LSH) per la ricerca di vicini approssimati, random projection per il kernel approximation, e la compressione delle rappresentazioni nei sistemi di recupero dell'informazione.

Lo Sketching estende queste idee all'algebra lineare numerica: invece di moltiplicare la matrice $A \in \mathbb{R}^{n\times d}$ con la matrice di osservazione completa, si moltiplica con una matrice "sketch" $S \in \mathbb{R}^{k\times n}$ con $k \ll n$, riducendo il problema $n \times d$ a uno $k \times d$ risolvibile in tempo $O(kd)$ invece di $O(nd)$. Il CountSketch usa righe con entrate $\{-1, 0, +1\}$ sparse — ogni colonna di A contribuisce a una sola riga dello sketch — permettendo il calcolo in tempo $O(\text{nnz}(A))$ (numero di elementi non nulli). Applicato alla regressione dei minimi quadrati, il sketching produce la soluzione approssimata $\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \|SA\mathbf{x}-S\mathbf{b}\|^2$ con errore relativo $(1+\varepsilon)$ rispetto alla soluzione ottimale usando $k = O(d/\varepsilon^2)$ righe dello sketch — una riduzione del problema da $n \times d$ a $d/\varepsilon^2 \times d$ indipendente da n.

CMA-ES: Evolution Strategy con Adattamento della Matrice di Covarianza

L'ottimizzazione di funzioni black-box tramite una distribuzione gaussiana adattiva — il principio del massimo passo, la regola di evoluzione cumulativa e la convergenza teorica.

368.1 · Il Problema dell'Ottimizzazione Black-Box

Molti problemi pratici di ottimizzazione non forniscono informazioni sul gradiente della funzione obiettivo: simulatori fisici, funzioni di valutazione di videogiochi, ottimizzazione di prompt discreti, progettazione di materiali tramite esperimento. In questi contesti, solo il valore della funzione (e talvolta nemmeno quello in modo deterministico) è accessibile — si parla di ottimizzazione *black-box* o *senza gradiente*. La Bayesian Optimization (§321) è efficace per funzioni molto costose con poche

valutazioni disponibili; gli algoritmi evolutivi sono preferibili quando si possono eseguire migliaia o milioni di valutazioni e la funzione obiettivo è relativamente economica da calcolare.

Il CMA-ES (Covariance Matrix Adaptation Evolution Strategy) (Hansen e Ostermeier 2001, con la versione moderna CMA-ES2001 di Hansen 2001) è l'algoritmo evolutivo di stato dell'arte per l'ottimizzazione di funzioni continue non convesse. L'idea fondamentale è di mantenere e adattare una distribuzione gaussiana $\mathcal{N}(\mathbf{m}, \sigma^2 C)$ nello spazio dei parametri, dove $\mathbf{m} \in \mathbb{R}^d$ è la media (il punto migliore corrente), $\sigma > 0$ è il passo globale, e $C \in \mathbb{R}^{d \times d}$ è la matrice di covarianza che cattura le correlazioni tra coordinate. Ad ogni generazione, si campionano λ individui $\mathbf{x}_k \sim \mathcal{N}(\mathbf{m}, \sigma^2 C)$, si valuta la funzione obiettivo su ciascuno, si selezionano i $\mu < \lambda$ migliori, e si aggiornano $\mathbf{m}$, σ, e C in modo da aumentare la probabilità dei punti buoni nelle generazioni successive. La chiave dell'efficienza di CMA-ES è l'adattamento intelligente della matrice di covarianza: essa apprende automaticamente la geometria locale della funzione obiettivo — le direzioni di maggiore miglioramento — permettendo di fare passi più grandi nelle direzioni vantaggiose e più piccoli in quelle sfavorevoli, senza calcolare alcun gradiente.

368.2 · L'Evoluzione Cumulativa del Percorso

L'aggiornamento della matrice di covarianza in CMA-ES usa la tecnica dell'evoluzione cumulativa del percorso (cumulative step-size adaptation): si mantiene un vettore di percorso $\mathbf{p}_c$ che accumula la direzione dei passi successivi nel tempo, e si aggiorna C in modo da aumentare la varianza nelle direzioni percorse ripetutamente. L'idea intuitiva è che se l'algoritmo continua a fare passi nella stessa direzione, significa che quella direzione è vantaggiosa e la distribuzione dovrebbe essere allungata in quella direzione. Formalmente: $\mathbf{p}_c \leftarrow (1 - c_c)\mathbf{p}_c + \sqrt{c_c(2 - c_c)\mu_{\text{eff}}}\,\frac{\mathbf{m}_{\text{new}} - \mathbf{m}}{\sigma}$ e $C \leftarrow (1 - c_1 - c_\mu)C + c_1 \mathbf{p}_c \mathbf{p}_c^T + c_\mu \sum_{k=1}^{\mu} w_k \mathbf{y}_k \mathbf{y}_k^T$ dove $\mathbf{y}_k$ sono i vettori di spostamento dei migliori individui normalizzati per σ. Il parametro c_c controlla la memoria dell'evoluzione cumulativa — valori alti danno più peso alla storia recente. CMA-ES scala a problemi con centinaia di dimensioni con qualità notevole, ed è lo strumento di scelta per l'ottimizzazione di parametri in simulatori di ingegneria, robotica soft, e il fine-tuning di iperparametri in regime ad alto budget.

OpenAI-ES e Neuroevoluzione Scalabile

Evolution Strategies su larga scala distribuita — gradiente gaussiano, confronto con i metodi di policy gradient, e i risultati su Mujoco e Atari.

369.1 · Evolution Strategies come Stima del Gradiente

Salimans, Ho, Chen, Sidor e Sutskever (OpenAI 2017) hanno riscoperto le Evolution Strategies come alternativa scalabile ai metodi di policy gradient per l'apprendimento per rinforzo. La loro analisi chiave è che lo stimatore di gradiente di un'Evolution Strategy gaussiana è matematicamente equivalente allo stimatore REINFORCE applicato alla distribuzione delle politiche: se si campionano perturbazioni $\boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, I)$ e si valuta la funzione obiettivo con parametri $\boldsymbol{\theta} + \sigma\boldsymbol{\epsilon}_i$, il gradiente stimato è:

$$\nabla_\theta J(\theta) \approx \frac{1}{n\sigma} \sum_{i=1}^{n} F(\theta + \sigma\boldsymbol{\epsilon}_i)\, \boldsymbol{\epsilon}_i$$

Questo è precisamente il gradiente gaussiano del payoff atteso sotto la distribuzione delle politiche $\pi_\theta = \mathcal{N}(\theta, \sigma^2 I)$ — la forma più semplice di policy gradient. Il vantaggio rispetto ai metodi RL standard è la parallelizzabilità estrema: ogni valutazione $F(\theta + \sigma\boldsymbol{\epsilon}_i)$ è indipendente dalle altre e può essere eseguita su un worker separato. Con n workers paralleli, si stima il gradiente in esattamente il tempo di una singola valutazione. OpenAI ha dimostrato che con 1440 cores paralleli, ES converge su task Mujoco di locomotione in pochi minuti invece delle ore richieste da PPO su un singolo core — una scalabilità quasi-lineare nel numero di workers. L'assenza di backpropagation — che richiede la memorizzazione delle attivazioni intermedie della rete e la loro comunicazione tra workers — rende ES particolarmente vantaggioso per reti neurali molto profonde su hardware distribuito dove la comunicazione del gradiente è un collo di bottiglia.

369.2 · Natural Evolution Strategies e il Gradiente Naturale per ES

Le Natural Evolution Strategies (NES) (Wierstra et al. 2014) applicano il concetto di gradiente naturale (§287) al contesto delle Evolution Strategies: invece di aggiornare i parametri della distribuzione di ricerca nella direzione del gradiente euclideo del fitness atteso, si usa il gradiente naturale rispetto alla metrica Fisher della distribuzione stessa. Per una distribuzione di ricerca gaussiana isotropa

$\mathcal{N}(\boldsymbol{\mu}, \sigma^2 I)$, il gradiente naturale del fitness si riduce a un aggiornamento normalizzato per la varianza corrente: $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} + \eta\sigma \sum_i w_i \boldsymbol{\epsilon}_i$ dove i pesi w_i sono derivati dal *rango* degli individui (fitness shaping) piuttosto che dal loro valore assoluto di fitness. Questa trasformazione monotona del segnale di fitness rende l'algoritmo invariante alle riscalature della funzione obiettivo, migliorando la robustezza su funzioni con scale molto diverse. La connessione con CMA-ES è profonda: CMA-ES può essere derivato come l'aggiornamento di gradiente naturale della distribuzione gaussiana anisotropa $\mathcal{N}(\mathrm{m}, \sigma^2 C)$ rispetto ai parametri (m, σ, C), con l'adattamento della matrice di covarianza che emerge naturalmente dall'aggiornamento del secondo momento del gradiente. OpenAI-ES è quindi una versione isotropa del NES, che sacrifica l'adattamento della curvatura in cambio di una parallelizzazione più semplice e scalabile.

Algoritmi Genetici e NEAT per la Neuroevoluzione

L'evoluzione della topologia delle reti neurali — crossover di strutture, speciation e l'approccio evolutivo alla ricerca di architetture.

370.1 · NEAT: Neuroevoluzione di Topologie Aumentanti

Gli algoritmi genetici classici per le reti neurali si concentrano sull'evoluzione dei pesi a topologia fissa. Il NEAT (NeuroEvolution of Augmenting Topologies) (Stanley e Miikkulainen 2002) evolve sia i pesi che la struttura della rete, permettendo l'emergenza di architetture complesse a partire da reti molto semplici. Le innovazioni chiave di NEAT rispetto ai precedenti tentativi di neuroevoluzione della struttura sono tre. La prima è il meccanismo di *marcatura storica* (historical markings): ogni nuovo gene (connessione o nodo) riceve un numero identificativo globale univoco che ne traccia l'origine evolutiva, permettendo il allineamento corretto di genomi con strutture diverse durante il crossover — il problema del "abbinamento competitivo" che aveva reso difficile il crossover strutturale nei sistemi precedenti. La seconda è la speciazione: individui con strutture simili sono raggruppati in specie, e la competizione avviene principalmente all'interno delle specie — questo protegge le innovazioni strutturali nuove dall'estinzione immediata per competizione con individui già ottimizzati di struttura diversa. La terza è la complessificazione incrementale: si inizia con reti minimaliste (senza nodi nascosti) e

si aggiunge struttura solo quando porta un miglioramento, evitando la complessità inutile. NEAT ha trovato applicazione nel game playing (bipede locomotion, gioco di Othello) e nella ricerca di architetture neurali, anticipando di dieci anni i metodi moderni di Neural Architecture Search.

370.2 · HyperNEAT e la Ricerca di Architetture Neurali

Un'estensione di NEAT particolarmente influente è HyperNEAT (Stanley, D'Ambrosio e Gauci 2009), che usa la rete evoluta da NEAT (chiamata CPPN, Compositional Pattern-Producing Network) non come rete di controllo diretto ma come funzione di connettività che genera i pesi di una rete più grande: $w_{ij} = \text{CPPN}(x_i, y_i, x_j, y_j)$ dove (x_i, y_i) e (x_j, y_j) sono le coordinate geometriche dei neuroni i e j nello spazio del substrato. Questo permette a HyperNEAT di sfruttare la regolarità geometrica del problema — pattern simmetrici, periodicità spaziali — senza imporla esplicitamente. La connessione con la Neural Architecture Search (NAS) moderna è diretta: metodi come AmoebaNet (Real et al. 2019) usano algoritmi evolutivi per cercare nello spazio di architetture definito da operazioni primitive (convoluzioni, pooling, skip connections), con un meccanismo di selezione basato sulle prestazioni sul validation set. AmoebaNet ha superato i modelli progettati a mano su ImageNet nel 2018, poi superato a sua volta da metodi differenziabili come DARTS. L'evoluzione e i metodi differenziabili si complementano: gli algoritmi evolutivi esplorano meglio lo spazio discreto delle architetture (operazioni diverse, skip connections), mentre i metodi basati su gradiente ottimizzano efficientemente i pesi una volta fissata l'architettura. Il campo della *AutoML end-to-end* tenta di combinare i due approcci in un unico sistema che ottimizza sia l'architettura che i pesi, con recenti sviluppi che usano Large Language Models come ottimizzatori di architetture tramite prompting iterativo (OPRO, Yang et al. 2024).

Ottimizzazione Gradient-Free vs. Gradient-Based: un Confronto Teorico

Quando i metodi senza gradiente competono — complessità query, regimi di alta dimensione e funzioni discontinue.

371.1 · La Query Complexity e i Limiti Fondamentali

Dal punto di vista teorico, i metodi gradient-free soffrono di una maledizione fondamentale per l'ottimizzazione ad alta dimensione: il loro costo di query scala esponenzialmente con la dimensione per funzioni non strutturate. Nesterov e Spokoiny (2017) hanno dimostrato che per funzioni L-smooth in $\mathbb{R}^d$, qualunque algoritmo gradient-free richiede almeno $\Omega(d/\varepsilon^2)$ valutazioni per trovare un punto con gradiente di norma al più ε — contro $O(1/\varepsilon^2)$ per i metodi gradient-based. Questo gap di d è precisamente il costo di stimare il gradiente tramite differenze finite: stimate d derivate parziali richiede d valutazioni aggiuntive. Per problemi con $d = 10^6$ (reti neurali grandi), questo gap è devastante in assenza di struttura aggiuntiva. La Bayesian Optimization supera questo limite per funzioni a dimensione intriseca bassa (dove la variazione è concentrata in poche direzioni), ed ES lo supera per problemi dove il parallelismo massivo compensa il costo per iterazione. La regola pratica è: si usa gradient-based quando il gradiente è disponibile a basso costo (backpropagation); si usa Bayesian Optimization per poche valutazioni costose (≤ 1000) su problemi con struttura smooth; si usa ES/CMA-ES per molte valutazioni parallele su problemi non differenziabili o simulazioni fisiche; si usano gli algoritmi genetici quando la struttura discreta (topologia di rete, sequenza di istruzioni) è parte dello spazio di ricerca.

371.2 · Evolutionary RL e il Confronto Empirico con PPO

Il confronto diretto tra metodi evolutivi (ES) e metodi di policy gradient (PPO, SAC) su benchmark standardizzati ha prodotto risultati sorprendenti. Such et al. (OpenAI 2017) hanno mostrato che una semplice Evolution Strategy Gaussiana con σ-adattamento — senza alcuna struttura adattativa tipo CMA-ES — eguaglia o supera DQN e A3C su molti videogiochi Atari, consumando tempi di wall-clock simili grazie alla parallelizzazione massiva. Il vantaggio di ES aumenta su task dove il gradiente ha scarsa utilità: ambienti con reward molto sparse (dove pass dall'azione all'effetto è lungo e tortuoso), task di locomotione con dinamiche caotiche (dove piccole perturbazioni nei parametri producono grandi variazioni nella traiettoria), e ottimizzazione di controller per hardware robotico reale (dove la simulazione è imperfetta). Il Genetic Programming estende ulteriormente la neuroevoluzione allo spazio dei programmi: anziché evolvere pesi o topologie di reti neurali, si evolvono programmi in linguaggi di programmazione semplici o alberi di espressioni matematiche, permettendo la scoperta di algoritmi simbolici

compatti. AlphaTensor (Fawzi et al. Nature 2022), pur non essendo ES puro, usa RL combinato con tecniche isperate alla ricerca evolutiva per trovare algoritmi di moltiplicazione matriciale più efficienti degli algoritmi classici — un risultato che ha migliorato l'algoritmo di Strassen (1969) per la prima volta in cinquant'anni.

Kernel Methods per Dati Tabellari: XGBoost vs Kernel SVM vs Deep Learning

Un confronto teorico e pratico sui tre paradigmi per la regressione e classificazione su dati strutturati — quando scegliere cosa e perché.

388.1 · La Struttura dei Dati Tabellari e le Sue Implicazioni per la Scelta del Modello

I dati tabellari — record con feature eterogenee (numeriche, categoriali, ordinali) senza struttura spaziale o sequenziale — costituiscono la maggior parte dei dati aziendali e medici, eppure sono il dominio dove il deep learning mostra i guadagni più modesti rispetto ai metodi classici. Shwartz-Ziv e Armon (2022) hanno condotto un'analisi sistematica su 45 dataset pubblici, trovando che XGBoost supera le reti neurali dense su $\approx 60\%$ dei dataset e le pareggia su molti altri — un risultato che ha stimolato un ampio dibattito sui motivi teorici. La risposta prevalente coinvolge tre caratteristiche strutturali dei dati tabellari. Prima, la presenza di feature con importanze molto diverse (alcune feature dominano la predizione): le reti neurali con attivazioni smooth trattano tutte le feature in modo simile all'inizio del training, mentre gli alberi di decisione selezionano naturalmente le feature più informative tramite split ottimali. Seconda, le relazioni non uniformi: molte relazioni nei dati tabellari sono funzioni a gradino o discontinue (una soglia di reddito, una classe d'età), ben catturate dagli alberi ma richiedenti molti layer per le reti smooth. Terza, le feature categoriali ad alta cardinalità: le reti neurali richiedono embedding appresi (con il rischio di overfitting su categorie rare), mentre XGBoost con encoding ordinale o target encoding funziona bene fuori dalla scatola.

388.2 · Kernel SVM per Dati Tabellari e le Limitazioni del Kernel RBF

Il Kernel SVM con kernel RBF è teoricamente universal (può approssimare qualunque funzione continua), ma in pratica soffre di due limitazioni per i dati tabellari. Prima, la sensibilità alle scale delle feature: il kernel RBF $k(\mathbf{x}, \mathbf{x}') = \exp(-\|\mathbf{x} -$

$\mathbf{x}'\|^2/(2\sigma^2))$ tratta tutte le dimensioni simmetricamente, quindi feature con scale diverse (reddito in migliaia di euro vs. età in anni vs. indicatori binari) devono essere normalizzate accuratamente — e la normalizzazione ottimale non è sempre ovvia. Seconda, il costo computazionale $O(n^3)$ in training e $O(n|SV|)$ in inferenza, che diventa proibitivo per $n > 10^5$. Il Kernel SVM con kernel di struttura ad albero (Milo et al. 2004) è un'alternativa interessante: il kernel $k(\mathbf{x}, \mathbf{x}')$ = (numero di foglie in comune nell'ensemble di alberi in cui entrambi i punti cadono) misura la similarità strutturale dei pattern di split degli alberi, ed è calcolabile in $O(T \log n)$ usando gli alberi dell'ensemble. Questo kernel combina l'espressività degli ensemble di alberi con il formalismo del kernel SVM, e la versione applicata a Random Forest produce il *forest kernel* — un kernel in cui la similarità dipende dalla frequenza con cui due punti finiscono nella stessa foglia attraverso tutti gli alberi della foresta.

Gaussian Process Latent Variable Models

GP come modelli generativi non lineari per la riduzione della dimensionalità — GPLVM, Bayesian GPLVM e le applicazioni alla visualizzazione di dati ad alta dimensione.

389.1 · GPLVM: Inversione del GP per la Riduzione Dimensionale

Il Gaussian Process Latent Variable Model (GPLVM) (Lawrence 2004) è la versione non lineare probabilistica della PCA: invece di proiettare dati ad alta dimensione in uno spazio a bassa dimensione tramite una trasformazione lineare, si usa un GP come funzione di decoder non lineare e si ottimizzano le coordinate latenti come variabili latenti. Formalmente, si osservano n punti $\mathbf{Y} \in \mathbb{R}^{n \times D}$ ad alta dimensione, si ipotizzano coordinate latenti $\mathbf{X} \in \mathbb{R}^{n \times q}$ a bassa dimensione, e si modella $\mathbf{Y} = f(\mathbf{X}) + \boldsymbol{\varepsilon}$ dove f è un GP. La log-likelihood del modello rispetto alle coordinate latenti è:

$$\log P(\mathbf{Y}|\mathbf{X}) = -\frac{D}{2}\log|K_{XX}| - \frac{1}{2}\mathrm{tr}(K_{XX}^{-1}\mathbf{Y}\mathbf{Y}^T) + \text{const}$$

dove K_{XX} è la matrice di covarianza del GP nelle coordinate latenti. L'ottimizzazione rispetto a $\mathbf{X}$ trova le coordinate latenti che massimizzano la plausibilità dei dati sotto il modello GP — equivalente a trovare la struttura a bassa dimensione nei dati

che è meglio spiegata da una funzione smooth GP. A differenza della PCA, GPLVM può catturare strutture non lineari (curve, tori, cluster) nelle coordinate latenti, e a differenza di t-SNE o UMAP fornisce un modello generativo che permette di campionare nuovi punti e di proiettare nuovi dati fuori dal training set. Il costo $O(n^3)$ in training è ridotto dalla versione sparsa (Sparse GPLVM) con inducing points — la stessa tecnica di §319 applicata alle coordinate latenti.

389.2 · Bayesian GPLVM e l'Inferenza Variazionale sulle Coordinate Latenti

Il GPLVM standard ottimizza le coordinate latenti come parametri puntuali — un approccio che sovra-adatta facilmente in regimi a basso numero di osservazioni per dimensione latente. Il Bayesian GPLVM (Titsias e Lawrence 2010) tratta le coordinate latenti come variabili aleatorie con prior gaussiana e inferisce la loro distribuzione posteriore tramite inferenza variazionale. La distribuzione variazionale $q(\mathbf{X}) = \prod_n \mathcal{N}(\mathbf{x}_n; \boldsymbol{\mu}_n, \mathbf{S}_n)$ è fattorizzata sugli individui (mean field), e i parametri $\boldsymbol{\mu}_n, \mathbf{S}_n$ vengono ottimizzati insieme agli iperparametri del GP massimizzando l'ELBO. Il Bayesian GPLVM è particolarmente utile per dati temporali dove le coordinate latenti hanno struttura temporale (il prior sulle coordinate può essere un GP temporale), per dati mancanti (la posterior sulle coordinate mancanti si integra naturalmente nell'ELBO), e per l'identificazione di strutture intrinseche nei dati biologici come l'analisi delle componenti latenti dell'espressione genica durante lo sviluppo cellulare.

Sparse Recovery e Compressed Sensing

Acquisizione e ricostruzione di segnali sparsi — il teorema RIP, LASSO come convex relaxation di ℓ^0, e le connessioni con l'inferenza bayesiana sparsa.

390.1 · Il Problema della Ricostruzione Sparsa

Il Compressed Sensing (Candès, Romberg e Tao 2006; Donoho 2006) ha rivoluzionato il campionamento dei segnali con il risultato controintuitivo che un segnale sparso — con solo $k \ll d$ componenti non nulle — può essere esattamente ricostruito da $m = O(k \log(d/k))$ misure lineari casuali, molto meno delle d misure richieste dal teorema di Nyquist-Shannon per segnali generici. Formalmente, si osserva $\mathbf{y} = \Phi \mathbf{x}^* + \boldsymbol{\varepsilon} \in \mathbb{R}^m$ con $m \ll d$, e si vuole recuperare $\mathbf{x}^* \in \mathbb{R}^d$ noto essere

k-sparso. Il problema di minimizzazione della ℓ^0-pseudo-norma $\min \|\mathbf{x}\|_0$ soggetto a $\Phi\mathbf{x} \approx \mathbf{y}$ è NP-difficile (combinatoriale). La sua rilassamento convesso LASSO $\min \|\mathbf{x}\|_1$ soggetto a $\Phi\mathbf{x} \approx \mathbf{y}$ è risolvibile in tempo polinomiale tramite programmazione lineare, e produce la stessa soluzione sparsa sotto la condizione RIP (Restricted Isometry Property): la matrice Φ soddisfa RIP di ordine k con costante δ_k se $(1-\delta_k)\|\mathbf{z}\|^2 \leq \|\Phi\mathbf{z}\|^2 \leq (1+\delta_k)\|\mathbf{z}\|^2$ per tutti i vettori k-sparsi $\mathbf{z}$. Matrici casuali gaussiane o bernulliane soddisfano RIP con alta probabilità per $m = O(k \log(d/k))$ — il punto chiave che giustifica la riduzione del numero di misure. Le applicazioni spaziano dall'imaging medico (dove ridurre le misure MRI significa ridurre il tempo in macchina) alla compressione di segnali radar, dall'acquisition di segnali sismici ai sistemi di raccomandazione con dati sparsi.

390.2 · Connessioni con il Machine Learning: LASSO, Elastic Net e Spike-and-Slab

Il LASSO (Tibshirani 1996) come regolarizzazione ℓ^1 nella regressione è il ponte tra il compressed sensing e il machine learning supervisionato: $\hat{\boldsymbol{\beta}} = \arg\min \|\mathbf{y} - X\boldsymbol{\beta}\|^2 + \lambda\|\boldsymbol{\beta}\|_1$ produce soluzioni sparse (molti coefficienti esattamente zero) grazie alla geometria della palla ℓ^1 — la funzione obiettivo tende a toccare la palla ℓ^1 agli spigoli, dove coordinate intere sono zero. Il percorso di regolarizzazione LASSO (Efron et al. 2004, LARS algorithm) calcola efficientemente l'intera sequenza di soluzioni per tutti i valori di λ, rivelando l'ordine in cui le feature entrano nel modello al decrescere della penalità — una forma di feature selection ordinata per importanza. L'Elastic Net (Zou e Hastie 2005) combina ℓ^1 e ℓ^2: $\lambda_1\|\boldsymbol{\beta}\|_1 + \lambda_2\|\boldsymbol{\beta}\|^2$, superando il LASSO in presenza di feature altamente correlate (dove il LASSO seleziona arbitrariamente una sola feature dal gruppo correlato, mentre l'Elastic Net le seleziona tutte con pesi simili). La versione bayesiana del LASSO è il prior di Laplace su $\boldsymbol{\beta}$: $P(\boldsymbol{\beta}|\lambda) \propto \prod_j e^{-\lambda|\beta_j|}$, e la MAP sotto questo prior coincide con la soluzione LASSO. La versione bayesiana più espressiva è il prior Spike-and-Slab (Mitchell e Beauchamp 1988): ogni coefficiente è una miscela di una delta in zero (lo spike) e una gaussiana larga (la slab), con probabilità di appartenenza alla componente non-zero che deve essere inferita. Lo Spike-and-Slab produce selezione di feature incertezza-calibrata e permette l'inferenza bayesiana completa sull'insieme di feature rilevanti, ma il costo computazionale esponenziale in d lo rende praticabile solo con approssimazioni variazionali o MCMC efficienti.

NOTEARS: Causal Discovery come Ottimizzazione Continua

La trasformazione del problema combinatorio del DAG learning in un problema di ottimizzazione smooth — il vincolo di aciclicità differenziabile e i suoi sviluppi.

372.1 · Il Problema Combinatorio della Struttura DAG

Il causal discovery classico (§303–304) usa test di indipendenza condizionale per costruire il grafo causale — un approccio che scala male ad alta dimensione perché il numero di test cresce esponenzialmente e i test non parametrici sono difficili in dimensione alta. Un'alternativa è il *score-based causal discovery*: si definisce uno score di bontà di adattamento per ogni possibile DAG (ad esempio, la log-likelihood penalizzata per la complessità strutturale), e si cerca il DAG che massimizza lo score. Il problema è che lo spazio dei DAG su d nodi ha dimensione super-esponenziale: il numero di DAG su d nodi cresce più velocemente di $2^{d^2/4}$, rendendo la ricerca esaustiva impraticabile già per $d = 20$. I metodi classici usano ricerca greedy (aggiunta/rimozione/inversione di archi) con ottimalità locale ma senza garanzie globali.

NOTEARS (NO TEARS: Non-combinatorial Optimization via Trace Exponential and Augmented lagrangian for Structure learning) (Zheng, Aragam, Ravikumar e Xing, NeurIPS 2018) ha proposto una soluzione elegante a questo problema che ha rivoluzionato il causal discovery: trasformare il problema combinatorio della ricerca del DAG ottimale in un problema di ottimizzazione continua tramite una caratterizzazione differenziabile della vincolo di aciclicità. La chiave è la seguente identità: una matrice di adiacenza pesata $W \in \mathbb{R}^{d \times d}$ corrisponde a un DAG se e solo se la matrice $e^{W \odot W}$ (esponenziale matriciale della matrice degli elementi al quadrato) ha traccia uguale a d, ovvero:

$$h(W) = \operatorname{tr}(e^{W \odot W}) - d = 0 \iff \text{il grafo corrispondente a } W \text{ è un DAG}$$

La funzione $h(W)$ è differenziabile rispetto a W (il gradiente è $\nabla h = (e^{W \odot W})^T \odot (2W)$, calcolabile in tempo $O(d^3)$), permettendo di usare metodi del Lagrangiano aumentato per ottimizzare lo score strutturale soggetto al vincolo $h(W) = 0$. Il problema di ottimizzazione risultante è continuo, smooth, e risolvibile con metodi di gradiente standard su dataset di dimensioni moderate. NOTEARS ha aperto

la strada a un'intera famiglia di metodi di causal discovery "gradient-based" che estendono il framework a DAG nonlineari (DAG-GNN), dati misti categoriali-continui, e setting causali con variabili latenti.

DAG-GNN e Score-Based Causal Discovery Non-Lineare

Estendere NOTEARS ai meccanismi causali non lineari — Graph Neural Networks per la struttura causale e i limiti dell'identificabilità.

373.1 · Dal Lineare al Non-Lineare: DAG-GNN

NOTEARS assume un modello causale lineare: $X_i = \sum_j W_{ji} X_j + \varepsilon_i$ — i genitori di ogni variabile influenzano linearmente la variabile figlia. Questa assunzione è spesso irrealistica per dati biologici (espressione genica, reti metaboliche) o economici dove le relazioni causali sono fortemente non lineari. DAG-GNN (DAG with Graph Neural Networks) (Yu, Chen, Gao e Yu, NeurIPS 2019) estende NOTEARS ai meccanismi causali non lineari utilizzando reti neurali — specificamente GNN — per parametrizzare le funzioni strutturali: $X_i = f_i(\mathbf{X}_{\mathbf{pa}(i)}) + \varepsilon_i$ dove f_i è una rete neurale. L'architettura usa encoder e decoder GNN per apprendere sia la struttura del grafo (la matrice di adiacenza W) sia i meccanismi causali (le funzioni f_i) in modo end-to-end, mantenendo il vincolo di aciclicità differenziabile di NOTEARS. Il training minimizza una combinazione di: ricostruzione dell'input (i dati osservati devono essere spiegabili dal modello causale), il vincolo di aciclicità $h(W) = 0$, e una penalità di sparsità sulla struttura del grafo (poiché il grafo causale reale è tipicamente sparso). DAG-GNN ha dimostrato una generalizzazione eccellente a reti causali reali da dati biologici, superando i metodi score-based lineari e i metodi constraint-based ad alta dimensione.

373.2 · Identificabilità dei Modelli Non Lineari e il Caso LiNGAM

Un problema fondamentale del causal discovery non lineare è la *score equivalence*: DAG con strutture diverse ma stessa Markov equivalence class producono la stessa distribuzione osservazionale, e non possono essere distinti senza assunzioni aggiuntive sulla forma funzionale. La notevole eccezione sono i modelli lineari non gaussiani: il modello LiNGAM (Linear Non-Gaussian Acyclic Model) (Shimizu et al. 2006) dimostra che se i meccanismi causali sono lineari ma le cause sono

non gaussiane, il DAG causale è identificabile univocamente — non solo fino alla Markov equivalence class. La ragione è il teorema di Darmois-Skitovitch: solo la distribuzione gaussiana è chiusa per combinazioni lineari, quindi per dati non gaussiani la direzione causale lineare si identifica tramite la non-gaussianità dei residui. Il metodo DirectLiNGAM (Shimizu et al. 2011) risolve il causal discovery lineare non gaussiano in tempo polinomiale con una sequenza di regressioni seguite da test di indipendenza sui residui. L'identificabilità non lineare generica, senza assunzioni sulla forma funzionale, rimane un problema aperto: è risolvibile con dati interventistici (sezione 375) oppure con il modello a rumore additivo (ANM), dove l'identificabilità segue dall'asimmetria della regressione nei due sensi causali.

Causal Representation Learning: Identità Causale delle Rappresentazioni

Apprendimento di rappresentazioni causalmente invarianti — disentanglement, Causal VAE e la connessione con IRM e DG.

374.1 · Il Problema del Disentanglement Causale

Il causal discovery classico assume che le variabili osservate siano le variabili causali rilevanti — ma in molte applicazioni del deep learning i dati sono pixel, token, o caratteristiche continue che non corrispondono direttamente alle cause del sistema. Il Causal Representation Learning (Schölkopf et al. 2021, come programma di ricerca) affronta il problema di apprendere rappresentazioni latenti che corrispondano alle variabili causali di un sistema, permettendo il trasferimento della conoscenza tra ambienti con interventi causali diversi. L'idea chiave è il principio di indipendenza causa-meccanismo: le cause di un sistema non contengono informazione sul meccanismo che genera gli effetti, e i meccanismi diversi (diversi layer del grafo causale) sono "modulari" — cambiarli indipendentemente non influenza gli altri. Questo principio, formalizzato da Peters, Janzing e Schölkopf (2017), è alla base dell'identificabilità: se si dispone di dati da più ambienti con diversi interventi, le variabili causali latenti possono essere identificate fino a una trasformazione nell'equivalence class dei modelli compatibili con tutti gli ambienti. In pratica, il Causal VAE e le sue varianti apprendono encoder che mappano le osservazioni in rappresentazioni latenti strutturate causalmente, dove diverse direzioni nello spazio latente corrispondono a diverse cause — a differenza del VAE standard

dove le dimensioni latenti non hanno interpretazione causale garantita.

374.2 · Invariant Risk Minimization e la Generalizzazione fuori Distribuzione

L'Invariant Risk Minimization (IRM) (Arjovsky, Bottou, Gulrajani e Lopez-Paz 2019) affronta la generalizzazione out-of-distribution da una prospettiva causale: si cerca una rappresentazione $\Phi(x)$ tale che il classificatore ottimale su $\Phi(x)$ sia lo stesso in tutti gli ambienti di training — una forma di invarianza che, per i modelli causali, è soddisfatta dalla rappresentazione delle cause dirette del target. La formalizzazione impone che la stessa funzione decisionale w debba essere simultaneamente ottimale su ogni ambiente $e \in \mathcal{E}$; nella versione pratica IRMv1 questo vincolo si rilassa a una penalite differenziabile che misura il gradiente residuo rispetto a w per ogni ambiente. L'intuizione causale è che predittori basati su correlazioni spurie — associazioni che variano tra ambienti — non possono soddisfare l'invarianza, mentre le caratteristiche causali producono per definizione la stessa funzione decisionale ovunque. Nella pratica, il benchmark DomainBed (Gulrajani e Lopez-Paz 2021) ha mostrato che IRM supera la minimizzazione del rischio empirico standard su alcuni dataset di domain generalization, ma non sistematicamente su tutti — a indicare che l'invarianza causale è condizione necessaria ma non sufficiente per la robustezza out-of-distribution quando il numero di ambienti di training è piccolo rispetto alla complessità delle correlazioni spurie presenti.

Causal Discovery con Dati Interventistici Misti

Integrare dati osservazionali e sperimentali per il recovery del DAG — l'algoritmo GIES, l'equivalenza interventistica e le applicazioni in genomica.

375.1 · La Forza degli Interventi per l'Identificabilità

Come discusso nella §302, i dati puramente osservazionali identificano il grafo causale solo fino alla Markov equivalence class — più DAG diversi sono compatibili con le stesse indipendenze condizionali. Disporre di dati interventistici — osservazioni dopo interventi $do(X_i = x)$ su alcune variabili — riduce l'ambiguità: ciascun intervento "taglia" gli archi entranti nella variabile manipolata, escludendo i DAG che non sono compatibili con la distribuzione interventistica. Il Greedy Interventional Equivalence Search (GIES) (Hauser e Bühlmann 2012) generalizza

l'algoritmo GES (Chickering 2002) per la ricerca score-based di DAG al caso di dati misti: si definisce l'equivalenza interventistica tra DAG come la compatibilità con tutte le distribuzioni osservazionali e interventistiche disponibili, e si ricerca il CPDAG interventistico (I-CPDAG) che rappresenta questa equivalenza class. L'applicazione più importante è la genomica perturbazionale: esperimenti CRISPR-Cas9 che silenziamo singoli geni in centinaia di cellule diverse forniscono dati interventistici su larga scala, permettendo di ricostruire le reti di regolazione genica con un livello di dettaglio molto superiore a quanto possibile con soli dati di espressione genica. Il progetto Perturb-seq e i dataset Replogle et al. (2022) con knockdowns di 10.000 geni in cellule umane rappresentano il dataset interventistico più grande mai raccolto, e stimolare lo sviluppo di algoritmi di causal discovery scalabili a questa dimensione è uno dei problemi aperti più attivi.

375.2 · Optimal Experimental Design per il Causal Discovery

Quando il budget sperimentale è limitato — come tipicamente avviene in biologia dove ogni esperimento di knockout genico costa tempo e denaro — la domanda naturale è: quali esperimenti interventistici scegliere per massimizzare l'informazione causale raccolta? Questo è il problema del Optimal Experimental Design (OED) per il Causal Discovery, o *Active Causal Learning*. La formalizzazione bayesiana considera la distribuzione a posteriori $P(\mathcal{G}|\mathcal{D})$ sul grafo causale $\mathcal{G}$ dati i dati $\mathcal{D}$ osservati finora, e cerca l'intervento I^* che massimizza il guadagno di informazione atteso: $I^* = rg\max_I \mathbb{E}_{y\sim P(y|I,\mathcal{D})}[\mathrm{KL}(P(\mathcal{G}|I,y,\mathcal{D})\|P(\mathcal{G}|\mathcal{D}))]$. Il guadagno di informazione massima corrisponde all'intervento che differenzia maggiormente le distribuzioni predette dai diversi DAG candidati — essenzialmente, esperimenti che "discriminano" al meglio tra grafi causali alternativi. Il calcolo esatto è intrattabile per spazi di grafi grandi, ma approssimazioni efficienti basate su sampling del posterior (MCMC su spazio di grafi), constraint propagation, e bandit bayesiani (che bilanciano exploitation del miglior intervento stimato con exploration di interventi incerti) hanno prodotto algoritmi di OED causale che riducono il numero di esperimenti necessari del 50-80% rispetto al campionamento random su benchmark di scoperta di reti regolatorie. L'integrazione di questi metodi con i grandi dataset di Perturb-seq è un campo di ricerca attivo che combina causal discovery, ottimizzazione bayesiana, e biologia computazionale.

Spazi di Probabilità e Misura: i Fondamenti Rigorosi

La costruzione matematica della probabilità su spazi generali — σ-algebre, misure di probabilità, integrazione di Lebesgue e variabili aleatorie.

376.1 · Perché la Teoria della Misura è Necessaria

Il machine learning opera quasi sempre con la probabilità intesa informalmente come "area sotto una curva" o "conteggio frequentista". Questa comprensione intuitiva è sufficiente per la pratica computazionale, ma diventa inadeguata non appena si vogliono enunciare teoremi di convergenza precisi — come la consistenza degli stimatori, la convergenza del MCMC, o la validità degli scambi limite-integrale nei bound variazionali. La teoria della misura di Lebesgue-Kolmogorov fornisce il fondamento rigoroso che permette di trattare la probabilità su spazi continui, spazi di funzioni, e spazi astratti arbitrari con la stessa algebra degli insiemi che si usa per gli spazi discreti. Senza questo fondamento, frasi come "la distribuzione degli embedding di una rete neurale infinitamente larga" o "la misura invariante di una catena di Markov su $\mathbb{R}^d$" non hanno significato preciso.

Uno spazio di probabilità è una tripla $(\Omega, \mathcal{F}, P)$ dove Ω è lo *spazio campionario* (l'insieme di tutti i possibili esiti), $\mathcal{F}$ è una *σ-algebra* su Ω (una famiglia di sottoinsiemi "misurabili" che include Ω, è chiusa per complementazione e per unioni numerabili), e $P : \mathcal{F} \to [0,1]$ è una *misura di probabilità* (una funzione σ-additiva con $P(\Omega) = 1$). La condizione di σ-additività — $P(\bigcup_n A_n) = \sum_n P(A_n)$ per insiemi disgiunti — è la versione insiemistica della proprietà di somma della probabilità, e la sua estensione agli insiemi numerabili (anziché solo finiti) è ciò che permette di passare dalla probabilità discreta alla continua. Una variabile aleatoria è una funzione misurabile $X : \Omega \to \mathbb{R}$ — misurabile nel senso che la controimmagine di ogni intervallo aperto appartiene a $\mathcal{F}$ — e l'integrale di Lebesgue $\mathbb{E}[X] = \int_\Omega X(\omega)\, dP(\omega)$ generalizza la somma pesata alla continua con la garanzia che l'integrale è ben definito per qualunque variabile aleatoria non negativa o integrabile.

376.2 · Misure su Spazi di Funzioni e Processi Stocastici

I Processi Gaussiani (§316) sono distribuzioni di probabilità su spazi di funzioni infinito-dimensionali — una generalizzazione diretta della distribuzione gaussiana da vettori finiti a funzioni continue. La teoria della misura fornisce la costruzione rigorosa di queste distribuzioni. Il teorema di estensione di Kolmogorov-Bochner

afferma che una famiglia consistente di distribuzioni finite-dimensionali $\{P_{t_1,\ldots,t_n}\}_{n,t_i}$ — consistente nel senso che le marginali delle distribuzioni più grandi coincidono con quelle delle distribuzioni più piccole — determina univocamente una misura di probabilità su uno spazio di funzioni appropriato. Per i GP, le distribuzioni finite-dimensionali sono le gaussiane multivariate con media e covarianza date dal kernel, e il teorema di Kolmogorov garantisce l'esistenza del processo come misura ben definita su $C(\mathcal{X})$ (funzioni continue su $\mathcal{X}$) sotto le opportune condizioni di regolarità del kernel. Senza questo risultato, il GP rimarrebbe solo una comoda notazione per distribuzioni finite-dimensionali consistenti, senza la garanzia di esistere come oggetto matematico unitario.

Leggi dei Grandi Numeri, TCL Funzionale e Convergenza degli Stimatori

I teoremi di convergenza fondamentali e le loro implicazioni per la consistenza degli algoritmi di machine learning.

377.1 · Le Leggi dei Grandi Numeri: Debole e Forte

La legge dei grandi numeri (LGN) è il risultato fondamentale che giustifica l'uso delle medie campionarie come stime delle aspettative. La LGN debole (Chebyshev 1866, per il caso i.i.d. con varianza finita) afferma che $\bar{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i \xrightarrow{P} \mathbb{E}[X]$ — convergenza in probabilità: per ogni $\varepsilon > 0$, $P(|\bar{X}_n - \mu| > \varepsilon) \to 0$. La LGN forte (Kolmogorov 1930, per il caso i.i.d. integrabile) garantisce la convergenza quasi certa: $\bar{X}_n \to \mathbb{E}[X]$ quasi certamente, cioè $P(\lim_{n\to\infty} \bar{X}_n = \mu) = 1$. La distinzione tra le due forme — convergenza in probabilità vs. quasi certa — è sottile ma cruciale: la convergenza in probabilità afferma che per ogni n grande la media è probabilmente vicina alla vera aspettativa; la convergenza quasi certa afferma che con probabilità uno la sequenza di medie converge, cioè che quasi tutti i percorsi campionari convergono. Per la statistica del ML, la LGN forte è il risultato rilevante: giustifica la sostituzione delle aspettative vere (intractable) con medie campionarie empiriche (computabili) nella derivazione degli stimatori di massima verosimiglianza, degli aggiornamenti di gradient descent stocastico, e dei bound di generalizzazione empirici.

377.2 · Il Teorema del Limite Centrale Funzionale

Il Teorema del Limite Centrale (TCL) classico afferma che $\sqrt{n}(\bar{X}_n - \mu) \xrightarrow{d} \mathcal{N}(0, \sigma^2)$ — la deviazione normalizzata della media campionaria converge in distribuzione a una gaussiana con varianza uguale alla varianza della singola variabile. Nel ML, il TCL giustifica l'uso delle gaussiane come distribuzione limite degli stimatori — ad esempio, la distribuzione dei gradienti stocastici in SGD per grandi batch, che motiva l'inizializzazione gaussiana e le analisi della stabilità del training. Il TCL funzionale (Teorema di Donsker) estende questa convergenza al processo campionario intero: il processo $\{B_n(t) = \sqrt{n}(\bar{X}_{[nt]} - \mu t/\sigma)\}_{t\in[0,1]}$ converge in distribuzione al moto browniano standard $\{W(t)\}_{t\in[0,1]}$ nello spazio $C([0,1])$ delle funzioni continue. Questa convergenza funzionale è alla base della teoria del limite per i bootstrap, delle proprietà asintotiche degli algoritmi di ottimizzazione stocastica a tempo continuo, e della connessione tra la traiettoria del gradient descent stocastico e le equazioni differenziali stocastiche di Langevin (§285).

Martingali e Disuguaglianze di Concentrazione

Sequenze con proprietà di "equità" — il teorema di convergenza per martingali, Azuma-Hoeffding e le disuguaglianze di Bernstein per la teoria del regret.

378.1 · Martingali: Sequenze senza Deriva

Un martingale è una sequenza di variabili aleatorie $\{M_t\}$ adattata a una filtrazione $\{\mathcal{F}_t\}$ che soddisfa la proprietà di equità: $\mathbb{E}[M_{t+1} \mid \mathcal{F}_t] = M_t$ — il valore atteso del termine successivo, data tutta la storia fino a t, è uguale al valore corrente. L'esempio canonico è il gioco equo: se un giocatore scommette equamente in ogni round, il capitale totale forma un martingale. Nel ML, i martingali appaiono in molteplici contesti. Le variazioni dell'errore di approssimazione nel SGD formano quasi-martingale sotto le opportune condizioni — proprietà che sta alla base delle prove di convergenza per SGD e Adam. I guadagni cumulativi degli algoritmi online (le somme parziali $\sum_t (f_t - \mathbb{E}[f_t \mid \mathcal{F}_{t-1}])$) sono differenze martingala quando le funzioni di loss sono casuali — permettendo di applicare le disuguaglianze di concentrazione per i martingali ai bound sul regret. Il teorema di convergenza per martingali bounded (Doob 1953) afferma che ogni martingale bounded converge quasi certamente — un risultato che applica direttamente alla convergenza del MCMC tramite

il teorema ergodico.

378.2 · Disuguaglianze di Azuma, Bernstein e McDiarmid

Le disuguaglianze di concentrazione per martingali sono i risultati più potenti per controllare le fluttuazioni delle medie campionarie e dei processi stocastici al di là di quanto possibile con la varianza sola. La disuguaglianza di Azuma-Hoeffding per martingale bounded: se $|M_t - M_{t-1}| \leq c_t$ quasi certamente, allora $P(M_n - M_0 \geq t) \leq \exp(-2t^2/\sum_k c_k^2)$ — la concentrazione gaussiana di una somma di variabili bounded, valida anche per sequenze dipendenti purché le differenze siano bounded. Questa disuguaglianza è lo strumento principale per i bound sul regret in OCO e bandit. La disuguaglianza di Bernstein affina Azuma aggiungendo l'informazione sulla varianza: $P(M_n \geq t) \leq \exp(-t^2/(2V + 2ct/3))$ dove $V = \sum_k \mathbb{E}[(M_k - M_{k-1})^2 \mid \mathcal{F}_{k-1}]$ è la varianza predittiva — il bound ha scala gaussiana $e^{-t^2/2V}$ per $t \ll V/c$ e scala esponenziale $e^{-t/c}$ per $t \gg V/c$. La disuguaglianza di McDiarmid è la versione "funzione stabile" di Hoeffding: se $f(x_1, \ldots, x_n)$ varia al più di c_i quando si cambia il i-esimo argomento, allora $P(f - \mathbb{E}[f] \geq t) \leq \exp(-2t^2/\sum_i c_i^2)$. McDiarmid è la base dei bound di generalizzazione in PAC learning: la differenza tra errore empirico e vero, vista come funzione del dataset, è stabile se si cambia un singolo esempio di training — e la costante di stabilità determina il tasso di concentrazione.

Convergenza dei Processi Stocastici e il Teorema Ergodico

La convergenza delle catene di Markov alla distribuzione stazionaria — mixing time, spettro della matrice di transizione e il teorema ergodico.

379.1 · Catene di Markov Ergodiche e la Distribuzione Stazionaria

La correttezza degli algoritmi MCMC — la garanzia che le medie temporali convergono alle aspettative sotto la distribuzione target — si fonda sul Teorema Ergodico per le Catene di Markov. Una catena di Markov discreta o continua con spazio degli stati $\mathcal{X}$ e nucleo di transizione $K(x, dy)$ è ergodica se è irriducibile (si può raggiungere qualunque stato da qualunque stato), aperiodica (non ha cicli), e positivamente ricorrente (il tempo di ritorno atteso a qualunque stato è finito). Il teorema ergodico afferma che per qualunque funzione integrabile f e

quasi ogni stato iniziale x_0, la media temporale converge all'aspettativa sotto la distribuzione stazionaria π: $\frac{1}{T}\sum_{t=1}^{T} f(X_t) \xrightarrow{T\to\infty} \int_{\mathcal{X}} f(x)\,\pi(dx)$ quasi certamente. La velocità di questa convergenza è determinata dal mixing time $t_{\text{mix}}(\varepsilon) = \min\{t : \max_{x_0} \|K^t(x_0,\cdot) - \pi\|_{TV} \le \varepsilon\}$ — il tempo necessario perché la distribuzione della catena sia ε-vicina alla distribuzione stazionaria in variazione totale, indipendentemente dal punto di partenza. Per catene reversibili su spazi discreti, il mixing time è legato al *spectral gap* $\gamma = 1 - \lambda_2$, dove λ_2 è il secondo autovalore della matrice di transizione: $t_{\text{mix}}(\varepsilon) = O(\gamma^{-1}\log(1/(\varepsilon\cdot\pi_{\min})))$ con $\pi_{\min} = \min_x \pi(x)$. Un piccolo spectral gap (autovalore vicino a 1) implica mixing lento — il problema tipico di Gibbs sampling su distribuzioni altamente correlate discusso nella §283.

379.2 · La Disuguaglianza di Poincaré e il Mixing delle SDE di Langevin

Per catene di Markov a spazio degli stati continuo — come la SGLD (§285) e l'HMC (§284) — il mixing time è caratterizzato dalla disuguaglianza di Poincaré per la distribuzione target π: $\text{Var}_\pi[f] \le C_P \mathbb{E}_\pi[\|abla f\|^2]$ per ogni funzione liscia f, dove C_P è la costante di Poincaré di π. Le distribuzioni fortemente log-concave $\pi \propto e^{-U}$ con $abla^2 U \succeq \mu I$ (cioè U è μ-fortemente convessa) soddisfano la disuguaglianza di Poincaré con $C_P \le 1/\mu$: la costante è tanto piccola quanto è forte la convessità, e il mixing è corrispondentemente veloce. Per la dinamica di Langevin continua $d\mathbf{z}_t = -abla U(\mathbf{z}_t)\,dt + \sqrt{2}\,d\mathbf{W}_t$, la convergenza esponenziale alla stazionarietà è garantita dalla disuguaglianza di Poincaré: $\text{KL}(p_t\|\pi) \le e^{-2\mu t}\text{KL}(p_0\|\pi)$ con tasso di convergenza 2μ per distribuzioni μ-fortemente log-concave. Per SGLD con gradiente stocastico, il risultato analogo introduce un errore stazionario proporzionale alla varianza del gradiente e al passo η: la catena converge non a π esatta ma a una distribuzione $\hat{\pi}$ con $\text{KL}(\hat{\pi}\|\pi) = O(\eta d/\mu^2)$ — un bias che scompare per $\eta o 0$ ma riappare per passi finiti, motivando i metodi di correzione di Metropolis-Hastings usati nelle versioni precondizionate di SGLD (SGHMC, SGLD-CV).

Teoria della Misura per le Reti Neurali Infinite

Il limite di larghezza infinita delle reti neurali — il Neural Tangent Kernel, la mean field theory e i processi gaussiani come limiti di reti profonde.

380.1 · La Rete Neurale come Processo Stocastico al Limite Infinito

Una delle connessioni più profonde tra la teoria della misura e il deep learning è emersa con la comprensione del comportamento delle reti neurali nel limite di larghezza infinita. Neal (1996) aveva osservato per primo che una rete neurale a un layer nascosto con pesi campionati i.i.d. da qualunque distribuzione con media zero e varianza finita converge, nel limite di larghezza infinita, a un Processo Gaussiano: le predizioni della rete su qualunque insieme finito di input sono distribuzioni gaussiane congiunte, con media zero e covarianza determinata dalla varianza dei pesi e dalla funzione di attivazione. Questo risultato è una conseguenza diretta del Teorema del Limite Centrale: la predizione della rete è una somma di n contributi indipendenti dai neuroni nascosti, e per $n \to \infty$ questa somma converge (per il TCL) a una gaussiana con media e varianza determinate dai momenti dei pesi. La varianza è il kernel neurale $k_{\text{NN}}(\mathbf{x}, \mathbf{x}') = \mathbb{E}_{\mathbf{w}}[\sigma(\mathbf{w}^T\mathbf{x})\sigma(\mathbf{w}^T\mathbf{x}')]$ — una media sulla distribuzione dei pesi del prodotto delle attivazioni — che dipende dalla funzione di attivazione σ e dalla distribuzione dei pesi.

380.2 · Neural Tangent Kernel e il Regime Lazy Training

Il risultato di Neal riguarda reti con pesi fissi campionati dalla distribuzione di inizializzazione — non reti addestrate. Il Neural Tangent Kernel (NTK) (Jacot, Gabriel e Hongler, NeurIPS 2018) estende questo risultato alle reti addestrate con gradient descent: nel limite di larghezza infinita, il training con gradient descent produce una rete la cui funzione è determinata da un kernel fisso — il NTK — che dipende solo dall'architettura e dall'inizializzazione, non dai dati di training. Il NTK è definito come $\Theta(\mathbf{x}, \mathbf{x}') = \nabla_\theta f(\mathbf{x}; \theta_0)^T \nabla_\theta f(\mathbf{x}'; \theta_0)$ — il prodotto interno dei gradienti della rete rispetto ai parametri, valutato all'inizializzazione. Nel regime di larghezza infinita, questo kernel rimane quasi costante durante il training (non si aggiorna con i pesi), e la rete si comporta come un modello lineare nello spazio delle feature implicite definito dal NTK. Questo regime, chiamato *lazy training* da Chizat, Oyallon e Bach (2019), è teoricamente trattabile ma probabilisticamente non è il regime in cui operano le reti neurali reali finite: le reti reali presentano *feature learning* (i neuroni cambiano le loro rappresentazioni durante il training) che è catturato solo da modelli teorici oltre il limite NTK, come la mean field theory delle reti neurali (Mei, Montanari e Nguyen 2018) che descrive l'evoluzione della distribuzione dei pesi tramite un'equazione di Fokker-Planck parziale — un'equazione differenziale sulla distribuzione dei parametri, non sui parametri stessi.

Causal Inference in Econometria: Differenze-in-Differenze e Regression Discontinuity

I metodi quasi-sperimentali per stimare effetti causali da dati osservazionali — l'assunzione di parallel trends, gli strumenti di Wald e il disegno a soglia.

391.1 · Differenze-in-Differenze: Sfruttare la Variazione Temporale

Il metodo delle Differenze-in-Differenze (DiD) è lo strumento causal inference più usato nell'econometria applicata quando si dispone di un "esperimento naturale": un evento esogeno che divide la popolazione in un gruppo trattato e un gruppo di controllo, e che produce variazione nel trattamento nel tempo. L'estimatore DiD calcola la differenza tra il cambiamento nel tempo del gruppo trattato e il cambiamento nel tempo del gruppo di controllo: $\hat{\tau}_{\text{DiD}} = (\bar{Y}_{T,1} - \bar{Y}_{T,0}) - (\bar{Y}_{C,1} - \bar{Y}_{C,0})$ dove i pedici T, C indicano trattamento/controllo e $0, 1$ indicano pre/post-intervento. L'assunzione fondamentale è il parallel trends: in assenza del trattamento, i trend temporali dei due gruppi sarebbero stati paralleli. Questa assunzione non è verificabile direttamente (non si osserva il controfattuale), ma si verifica indirettamente controllando che i trend pre-intervento siano effettivamente paralleli. Quando il trattamento viene assegnato in diversi momenti a diversi gruppi — il setting "staggered DiD" sempre più comune con dati di panel lunghi — la stima DiD classica può essere biasata perché usa trattati precoci come controlli per i trattati tardivi in momenti in cui il trattamento ha già effetti dinamici. I metodi recenti di Callaway-Sant'Anna (2021) e Sun-Abraham (2021) correggono questo bias con stimatori heterogeneity-robust che aggregano effetti causali per coorte e tempo di trattamento.

391.2 · Regression Discontinuity e Strumenti Naturali

Il Regression Discontinuity Design (RDD) sfrutta le discontinuità nelle regole di assegnazione del trattamento: se il trattamento è assegnato in base a una variabile di "running" che supera una soglia c (es. ammissione universitaria per chi supera un punteggio di ammissione), l'effetto causale si stima come la discontinuità nella variabile di outcome in c. Vicino alla soglia, le unità appena sopra e appena sotto sono quasi identiche in tutte le caratteristiche osservate e non osservate — una forma di randomizzazione locale. Il estimatore RDD locale lineare stima la discon-

tinuità $\hat{\tau} = \lim_{x \to c^+} \hat{f}(x) - \lim_{x \to c^-} \hat{f}(x)$ dove $\hat{f}$ è una regressione lineare locale stimata separatamente da sinistra e destra della soglia. La bandwidth ottimale bilancia il bias (usare osservazioni lontane dalla soglia che potrebbero avere caratteristiche diverse) e la varianza (usare poche osservazioni vicino alla soglia). Il metodo Instrumental Variables (IV) affronta il bias da variabili omesse tramite uno "strumento" Z — una variabile che influenza il trattamento D ma non l'outcome Y se non attraverso D. L'estimatore di Wald $\hat{\tau}_{\text{IV}} = \text{Cov}(Z, Y)/\text{Cov}(Z, D)$ stima l'effetto causale locale (LATE) per il sottogruppo di "compliers" — le unità che cambiano trattamento in risposta allo strumento. L'integrazione di questi metodi con il machine learning (Double/Debiased ML di Chernozhukov et al. 2018) permette di controllare per un numero elevato di confounders tramite modelli ML senza introdurre il bias di regolarizzazione nell'estimatore causale finale.

Conformal Prediction: Incertezza Senza Assunzioni Distribuzionali

Predizione con garanzie di copertura marginal-free — il metodo di Venn, la conformal prediction split e le applicazioni all'inferenza affidabile.

392.1 · Il Framework della Conformal Prediction

La Conformal Prediction (Vovk, Gammerman e Shafer 2005) affronta il problema della quantificazione dell'incertezza in modo radicalmente diverso da tutti i metodi bayesiani o frequentisti classici: produce insiemi di predizione con garanzia di copertura esatta e finitostica senza fare alcuna assunzione sulla distribuzione dei dati, tranne l'intercambiabilità (una versione più debole dell'i.i.d.). L'intuizione è semplice: dato un nuovo punto x_{n+1}, per ogni possibile etichetta candidata y si calcola una misura di "conformità" che quantifica quanto la coppia (x_{n+1}, y) si adatti al training set, e si include y nell'insieme di predizione se la sua conformità non è significativamente peggiore di quelle dei punti di training. Formalmente, la p-value di conformal per l'ipotesi $y_{n+1} = y$ è $p(y) = |\{i : \alpha_i \geq \alpha_{n+1}(y)\}|/(n+1)$ dove α_i sono le *nonconformity scores* — misure di quanto ogni punto di training è "strano" rispetto agli altri. L'insieme di predizione al livello $1 - \alpha$ è $\Gamma^\alpha(x_{n+1}) = \{y : p(y) > \alpha\}$. La garanzia fondamentale è che $P(y_{n+1} \in \Gamma^\alpha(x_{n+1})) \geq 1 - \alpha$ per qualunque distribuzione scambiabile dei dati — una garanzia di copertura marginale esatta, non approssimata, senza assunzioni parametriche.

392.2 · Split Conformal e Conformal Risk Control

La versione computazionalmente efficiente è la Split Conformal Prediction: si divide il training set in un insieme di training "vero" e un insieme di calibrazione, si addestra il modello sul primo e si calcolano le nonconformity scores sul secondo. La nonconformity score più comune per la regressione è il residuo assoluto: $\alpha_i = |y_i - \hat{f}(x_i)|$. L'insieme di predizione per un nuovo x_{n+1} è $\hat{f}(x_{n+1}) \pm q_{1-\alpha}$ dove $q_{1-\alpha}$ è il $(1-\alpha)$-quantile empirico dei residui di calibrazione. Questo produce intervalli di predizione esattamente calibrati con copertura $\lceil (1-\alpha)(m+1) \rceil / m \geq 1 - \alpha$ dove m è la dimensione del set di calibrazione. L'estensione più potente è il Conformal Risk Control (Angelopoulos et al. 2022) che generalizza la garanzia di copertura a qualunque rischio monotono nell'insieme di predizione: si può garantire che il rischio atteso del modello (accuratezza, falsi positivi, coverage semantica) sia al di sotto di una soglia specificata, senza assunzioni distribuzionali, su qualunque tipo di output (classificazione multi-label, segmentazione di immagini, traduzione automatica). Conformal Prediction è diventato il metodo di quantificazione dell'incertezza di riferimento nei sistemi di IA ad alta affidabilità dove le garanzie formali sono richieste — diagnosi medica, guida autonoma, trading algoritmico.

Fairness, Bias e Apprendimento Equo

Le definizioni matematiche di equità — disparate impact, equalized odds, calibrazione e i teoremi di impossibilità dell'equità simultanea.

393.1 · Le Definizioni Formali di Equità e i Loro Conflitti

L'equità algoritmica è diventata un campo matematicamente preciso con la formalizzazione di diverse nozioni di equità che un classificatore può (o non può) soddisfare simultaneamente. Sia Y il target vero, $\hat{Y}$ la predizione del modello, e A un attributo protetto (genere, etnia, ecc.). Le tre principali definizioni di equità sono:

La demographic parity (disparate impact): $P(\hat{Y} = 1 | A = 0) = P(\hat{Y} = 1 | A = 1)$ — la probabilità di previsione positiva è la stessa nei due gruppi. Questa definizione è soddisfatta da un classificatore che ignora completamente A, ma può essere ingiusta se i due gruppi hanno tassi di base diversi del target Y.

Le equalized odds (Hardt, Price e Srebro 2016): $P(\hat{Y} = 1|Y = y, A = 0) = P(\hat{Y} = 1|Y = y, A = 1)$ per $y \in \{0, 1\}$ — i tassi di falsi positivi e falsi negativi sono uguali nei due gruppi. Questa definizione condiziona sulla vera etichetta Y, ed è più esigente della demographic parity.

La calibrazione: $P(Y = 1|\hat{p}, A = 0) = P(Y = 1|\hat{p}, A = 1) = \hat{p}$ — le probabilità predette hanno lo stesso significato empirico nei due gruppi. Un classificatore calibrato è "onesto" nel senso che le sue probabilità riflettono le frequenze effettive, ma può avere tassi di errore diversi nei due gruppi se le distribuzioni di Y condizionalmente ad A sono diverse.

Il teorema di Chouldechova (2017) dimostra che demographic parity, equalized odds e calibrazione non possono essere soddisfatti simultaneamente (a meno di casi degeneri) quando le probabilità di base $P(Y = 1|A = 0) \neq P(Y = 1|A = 1)$ sono diverse. Questo risultato, noto come *impossibilità dell'equità simultanea*, implica che qualunque definizione di equità richiede scelte di valore che vanno oltre la matematica: non esiste un classificatore "intrinsecamente equo" che soddisfi tutte le definizioni contemporaneamente. La counterfactual fairness (Kusner et al. 2017, §307) offre un'alternativa causale: la predizione è equa se non cambierebbe se il gruppo di appartenenza dell'individuo fosse diverso, tenendo fissi tutti gli altri attributi causalmente indipendenti da A — una definizione che richiede un modello causale esplicito della struttura dei dati.

Teoria della Complessità per il Machine Learning

I limiti computazionali dell'apprendimento — apprendimento nei modelli circuit, hardness of learning e la connessione con la crittografia.

394.1 · Problemi NP-difficili nel Machine Learning

La teoria della complessità risponde a una domanda diversa ma complementare alla teoria statistica dell'apprendimento: non "quanti dati servono?" ma "quanto tempo computazionale serve?". Molti problemi centrali del ML sono NP-difficili nel senso del worst-case, il che implica che non esistono algoritmi efficienti per tutte le istanze (a meno che P=NP). L'addestramento di una rete neurale a due layer per minimizzare l'errore quadratico esatto è NP-difficile (Blum e Rivest 1992) — sebbene nella pratica il gradient descent su reti sovraparametrizzate trovi soluzioni ottime quasi

sempre, il caso worst-case è intrattabile. Il problema dell'ERM sull'intersezione di semispazi (trovare un'ipotesi in $\{h = \text{sign}(\sum_i \mathbf{1}[\mathbf{w}_i^T \mathbf{x} \geq 0])\}$ con minimo errore empirico) è NP-difficile già per 2 semispazi. Il problema del Planted Clique — trovare un clique di dimensione $k = O(\sqrt{n})$ in un grafo casuale $G(n, 1/2)$ con una clique piantata — è congetturalmente difficile per gli algoritmi a tempo polinomiale, e questa congettura è stata usata per dimostrare lower bound computazionali per problemi di statistica computazionale come il testing di clustering e il causal discovery a bassa dimensione.

394.2 · Apprendimento e Crittografia: Impossibilità Computazionale

Kearns e Valiant (1994) hanno dimostrato un risultato sorprendente: apprendere alcune classi di funzioni booleane semplici è computazionalmente equivalente a rompere algoritmi crittografici standard — se esistesse un algoritmo efficiente per apprendere quelle classi, si potrebbero invertire le funzioni one-way su cui si basa la sicurezza RSA e della crittografia a chiave pubblica. Specificamente, apprendere le funzioni parità sparse (*learning parity with noise*, LPN) è equivalente a rompere alcuni codici di correzione degli errori, e il problema è la base di sistemi crittografici post-quantum. Questo risultato ha implicazioni pratiche: certi problemi non saranno mai risolti da algoritmi di ML efficienti non perché mancano i dati, ma perché la loro soluzione computazionale è crittograficamente difficile. D'altra parte, la connessione tra ML e crittografia è fruttifera anche nella direzione opposta: i teoremi di sicurezza crittografica forniscono strumenti per dimostrare la *privacy* delle predizioni ML — i risultati fondamentali sulla privacy differenziale (§228) usano costruzioni di meccanismi casuali formalmente analoghi ai sistemi crittografici a chiave simmetrica.

Equivarianza e Reti Neurali con Simmetria

Il teorema di universalità equivariante, i gruppi di simmetria rilevanti per il ML e le architetture che li rispettano — dai transformer alle reti molecolari.

395.1 · Il Teorema di Cohen-Welling e l'Equivarianza come Vincolo Architetturale

Una delle intuizioni più produttive del deep learning moderno è che le architetture devono rispettare le simmetrie del problema: una rete che processa immagini

dovrebbe essere equivariante alla traslazione (spostare un oggetto nell'immagine sposta la sua rappresentazione, non la cambia qualitativamente), una rete che processa molecole dovrebbe essere equivariante alle rotazioni e alle riflessioni nello spazio 3D (due molecole identiche in orientamento diverso devono avere la stessa energia predetta). Il Teorema di Cohen e Welling (2016) fornisce la caratterizzazione completa delle funzioni lineari equivarianti rispetto all'azione di un gruppo G: ogni mappa lineare $\phi : \mathbb{R}^{|G|} \to \mathbb{R}^{|G|}$ equivariante rispetto all'azione regolare di G (shift delle coordinate per gli elementi del gruppo) può essere scritta come convoluzione sul gruppo. Per i gruppi continui compatti come SO(3), la decomposizione di Fourier sul gruppo usa le rappresentazioni irriducibili del gruppo come "frequenze", e la convoluzione equivariante è un prodotto di Clebsch-Gordan nella base delle rappresentazioni irriducibili — la struttura usata in e3nn (§311) per la chimica computazionale. La chiave pratica è che l'equivarianza riduce il costo effettivo dell'apprendimento: una rete equivariante non deve imparare separatamente come rispondere alla stessa molecola in 100 orientamenti diversi — la risposta a un orientamento determina automaticamente le risposte a tutti gli altri tramite la struttura del gruppo.

395.2 · Il Principio di Equivarianza nei Transformer

La self-attenzione dei Transformer è equivariante alla permutazione dell'input: scambiare due token produce lo stesso output ma con i token nell'ordine scambiato. Questa è la simmetria fondamentale dei dati insiemistici (set processing) — dove l'ordine degli elementi non ha significato intrinseco. Il positional encoding rompe questa simmetria aggiungendo informazione sull'ordine, ma la self-attenzione senza positional encoding è pienamente permutation-equivariante, il che la rende naturalmente applicabile a set di oggetti senza ordine canonico (molecole come set di atomi, scene come set di oggetti, grafi come set di nodi). Le Set Transformer e le Deep Sets (Zaheer et al. 2017) formalizzano questa osservazione: qualunque funzione di set permutation-invariante può essere scritta come $\rho(\sum_{x \in \mathcal{X}} \phi(x))$ per opportune funzioni ϕ e ρ (teorema di Zaheer et al.) — la self-attenzione seguita da aggregazione è un caso speciale con ϕ dipendente dal contesto. La connessione con la geometria dell'informazione è profonda: le famiglie esponenziali con parametri naturali additivi (gaussiane, Poisson, Bernoulli) hanno statistiche sufficienti che si aggregano per somma, rendendo la forma $\sum \phi(x_i)$ l'operazione fondamentale per le statistiche sufficienti di set — una connessione tra la struttura dei gruppi, le famiglie esponenziali e le architetture neurali che suggerisce principi di design uni-

ficati per il deep learning equivariante.

Score Matching e Modelli Basati sull'Energia

Stimare la densità senza normalizzazione — lo score della distribuzione, l'obiettivo di Hyvärinen e la connessione con i modelli di diffusione.

396.1 · Il Problema della Costante di Normalizzazione

Molti modelli probabilistici naturali hanno la forma $p_\theta(\mathbf{x}) = \exp(-E_\theta(\mathbf{x}))/Z(\theta)$ dove E_θ è una funzione di energia (non necessariamente limitata) e $Z(\theta) = \int \exp(-E_\theta(\mathbf{x}))\, d\mathbf{x}$ è la funzione di partizione — tipicamente intrattabile. I modelli di questo tipo, detti Energy-Based Models (EBM), includono le Boltzmann Machine (§278), i campi di Markov casuali (§277), e le distribuzioni di Gibbs per i sistemi fisici. L'addestramento tramite massima verosimiglianza richiede il calcolo di $\nabla_\theta \log Z(\theta) = -\mathbb{E}_{p_\theta}[\nabla_\theta E_\theta(\mathbf{x})]$ — un'aspettativa sotto il modello stesso, intrattabile senza MCMC. Il Contrastive Divergence (Hinton 2002) approssima questo gradiente con pochi passi di Gibbs sampling, ma produce stime biasate che possono ostacolare la convergenza.

396.2 · Score Matching: l'Obiettivo di Hyvärinen

Hyvärinen (2005) ha proposto un'alternativa elegante che evita completamente il calcolo di $Z(\theta)$: invece di stimare la densità p_θ, si stima il suo score — il gradiente del logaritmo della densità $\mathbf{s}_\theta(\mathbf{x}) = \nabla_\mathbf{x} \log p_\theta(\mathbf{x}) = -\nabla_\mathbf{x} E_\theta(\mathbf{x})$. Il punto chiave è che lo score non dipende da $Z(\theta)$ (la costante di normalizzazione si cancella nel gradiente). L'obiettivo di score matching misura la distanza Fisher tra lo score del modello e lo score dei dati veri:

$$\mathcal{J}_{\mathrm{SM}}(\theta) = \mathbb{E}_{p_{\mathrm{data}}}\left[\frac{1}{2}\|\mathbf{s}_\theta(\mathbf{x})\|^2 + \nabla_\mathbf{x} \cdot \mathbf{s}_\theta(\mathbf{x})\right]$$

La formula è notevole perché contiene solo aspettative rispetto ai dati — non rispetto al modello — e può essere stimata con semplici medie campionarie. Il termine $\nabla_\mathbf{x} \cdot \mathbf{s}_\theta = \sum_i \partial s_{\theta,i}/\partial x_i$ è la divergenza dello score, calcolabile tramite autodifferenziazione ma con costo $O(d)$ backward pass per ogni campione. Il Sliced Score

Matching (Song et al. 2019) riduce questo costo proiettando il vettore score su direzioni casuali, riducendo il costo a $O(1)$ backward pass con una piccola perdita di efficienza statistica. La connessione con i modelli di diffusione (score-based diffusion, Song e Ermon 2020) è diretta: DDPM e il suo omologo continuo apprende lo score della distribuzione dei dati perturbati con diversi livelli di rumore gaussiano, e la generazione avviene invertendo il processo di diffusione guidata dallo score appreso — un campionamento dalla distribuzione target tramite dinamica di Langevin guidata dallo score.

396.3 · MCMC basato sullo Score: Langevin e HMC come Caso Speciale

La dinamica di Langevin stocastica per campionare da $p \propto e^{-E}$ è esattamente un processo guidato dallo score: $d\mathbf{x}_t = -\nabla E(\mathbf{x}_t)\,dt + \sqrt{2}\,d\mathbf{W}_t = \mathbf{s}(\mathbf{x}_t)\,dt + \sqrt{2}\,d\mathbf{W}_t$. HMC estende questa dinamica in modo hamiltoniano per una migrazione più efficiente dello spazio. Se si dispone di uno score model addestrato $\mathbf{s}_\theta \approx \nabla \log p$, si può campionare direttamente dalla distribuzione target senza mai valutare E o Z, semplicemente sostituendo lo score vero con quello appreso. Questo è il principio dei modelli generativi score-based: si addestra uno score network su dati rumorosi con diversi livelli di rumore σ (usando il denoising score matching, che è equivalente allo score matching ma più stabile numericamente), poi si usa la Langevin annealed per campionare progressivamente dai livelli di rumore più alti ai più bassi — il processo di "denoising" che, nel limite del tempo continuo, corrisponde all'inversione stocastica della SDE di diffusione (la "reverse SDE" di Song et al. 2021). La notevole unificazione di DDPM, SMLD e altri modelli di diffusione come casi speciali della stessa SDE di diffusione nella formulazione di Song-Ermon-Kingma è uno dei risultati teorici più eleganti del deep learning recente.

Normalizing Flows Avanzati: RealNVP, Glow e Neural Spline Flows

Trasformazioni biettive con determinante efficiente — accoppiamento affine, convoluzioni 1×1 e la spline come soluzione al tradeoff espressività-efficienza.

397.1 · Il Change-of-Variables Formula e la Regola del Determinante

I Normalizing Flows costruiscono distribuzioni complesse come pushforward di distribuzioni semplici (tipicamente gaussiane) attraverso trasformazioni biettive

differenziabili $f_\theta : \mathbb{R}^d \to \mathbb{R}^d$. Se $\mathbf{z} \sim p_Z$ (la distribuzione base) e $\mathbf{x} = f_\theta(\mathbf{z})$, la densità dei dati è data dalla formula del cambio di variabile:

$$\log p_X(\mathbf{x}) = \log p_Z(f_\theta^{-1}(\mathbf{x})) + \log \left| \det \frac{\partial f_\theta^{-1}}{\partial \mathbf{x}} \right|$$

Il calcolo del determinante dello Jacobiano in $O(d^3)$ è proibitivo per reti di grandi dimensioni — da qui la necessità di costruzioni speciali con Jacobiani strutturati. Il RealNVP (Dinh, Sohl-Dickstein e Bengio 2017) usa le trasformazioni di accoppiamento affine: si dividono le coordinate in due blocchi $(\mathbf{x}_a, \mathbf{x}_b)$, il blocco a passa invariato, il blocco b viene scalato e traslato in funzione di a: $\mathbf{y}_b = \mathbf{x}_b \odot \exp(s(\mathbf{x}_a)) + t(\mathbf{x}_a)$. Il Jacobiano è triangolare a blocchi con diagonale $\exp(s(\mathbf{x}_a))$, quindi il determinante è $\sum_i s_i(\mathbf{x}_a)$ — calcolabile in $O(d)$. Alternando i ruoli dei due blocchi in layer successivi, si garantisce che tutte le dimensioni vengano trasformate.

397.2 · Glow: Convoluzioni 1×1 e Multiscala

Glow (Kingma e Dhariwal 2018) estende RealNVP con due innovazioni. Le convoluzioni 1×1 apprendibili sostituiscono il meccanismo di shuffle fisso delle dimensioni tra layer: ogni layer apprende una matrice di permutazione/rotazione $W \in \mathbb{R}^{c \times c}$ (dove c è il numero di canali) con $\log |\det W|$ come contributo al log-Jacobiano. Per mantenere efficiente il calcolo del determinante, W viene parametrizzata tramite decomposizione LU: $W = PL(U + \text{diag}(\mathbf{s}))$ con P una permutazione fissa, L triangolare inferiore con diagonale unitaria, U triangolare superiore con diagonale nulla, e $\mathbf{s}$ il vettore dei valori diagonali. Il determinante è allora $\prod_i s_i$ — calcolabile in $O(d)$. L'architettura multiscala elabora le immagini a risoluzioni diverse: a ogni livello si separa metà dei canali (inviandoli direttamente al decoder) e si elabora il resto — una struttura piramidale che riduce il costo computazionale da $O(d)$ a $O(d \log d)$ per immagine. Glow è stato il primo flow model a produrre immagini facciali fotorealistiche a 256×256 pixel.

397.3 · Neural Spline Flows: Monotone Transformations con Alta Capacità

I Neural Spline Flows (NSF) (Durkan, Bekasov, Murray e Papamakarios 2019) sostituiscono le trasformazioni affini nei coupling layer con funzioni più espressive ma ancora invertibili: funzioni monotone a tratti costruite come spline razionali quadratiche. Una spline razionale quadratica su $[-B, B]$ è definita da K punti di con-

trollo interni con posizioni e derivate positive apprendibili, che si interconnettono in una funzione monotona liscia con derivata ovunque positiva. Il determinante del Jacobiano della spline è la derivata della spline stessa — calcolabile in $O(K)$ per dimensione — e l'inversione avviene tramite la formula analitica per le spline razionali quadratiche (a differenza delle spline cubiche, che richiedono risoluzione numerica per l'inversione). NSF raggiunge log-likelihood superiori a RealNVP e Glow con lo stesso numero di layer, grazie alla maggiore capacità delle trasformazioni spline rispetto alle affini, e compete con i modelli autoregressivi (che sono più espressivi ma più lenti in generazione) su benchmark di modellazione della densità su dati tabulari e immagini a bassa risoluzione.

Teoria della Generalizzazione per i Transformer

Bound di generalizzazione specifici per l'architettura Transformer — la norma spettrale, la Rademacher complexity degli attention layers e il fenomeno della grokking.

398.1 · Norm-Based Bounds per i Transformer

I bound di generalizzazione classici basati sulla VC dimension (§354) sono troppo laschi per i Transformer — prevederebbero overfitting su qualunque dataset ragionevole. I bound norm-based (§355) sono più fini: Edelman et al. (2022) hanno derivato bound di Rademacher complexity per i Transformer che scalano con il prodotto delle norme spettrali dei layer di attention e FFN. Per un Transformer a L layer con norma spettrale dei pesi di attention bounded da β e norme dei vettori di input bounded da B, la Rademacher complexity scala come $O(B\beta^L\sqrt{d_{\text{model}}\log m/m})$ dove m è la dimensione del training set. Questo bound — come i bound norm-based per le reti fully connected — dipende dalla norma dei pesi, non dal numero di parametri: un Transformer con pesi piccoli può generalizzare bene anche con miliardi di parametri, coerente con l'osservazione empirica che i LLM moderni generalizzano notevolmente bene nonostante l'enorme numero di parametri. La conseguenza pratica è che tecniche che mantengono piccole le norme dei pesi durante il training — come weight decay, gradient clipping, e la normalizzazione dello spettro — hanno un fondamento teorico preciso nei bound di generalizzazione.

398.2 · Il Fenomeno della Grokking

Il fenomeno della grokking (Power et al. 2022) è uno dei risultati empirici più sorprendenti del deep learning recente: un Transformer addestrato su operazioni modulari (addizione mod p, per esempio) inizialmente memorizza il training set senza generalizzare, poi — dopo molte epoche di training *dopo* aver già azzerato il training loss — improvvisamente "capisce" l'algoritmo e la accuratezza di validazione salta da caso ($1/p$) a quasi perfetta. Questo comportamento — una generalizzazione ritardata e quasi discontinua — contraddice l'intuizione che l'overfitting e la generalizzazione siano in tradeoff immediato. La spiegazione prevalente è in termini di soluzione circuitale vs. memorizzazione: durante la prima fase il modello memorizza il training set con una soluzione non algoritmica (ad alta norma dei pesi); durante la seconda fase, guidato dalla pressione del weight decay che penalizza le norme elevate, il modello trova una soluzione algoritmica compatta (a bassa norma) che generalizza — una vera "comprensione" dell'operazione modulare rappresentata come circuito nella rete. La grokking è osservabile solo con weight decay non trascurabile: senza regolarizzazione, la soluzione ad alta norma è stabile e il modello rimane in overfitting indefinitamente. Questo suggerisce che la regolarizzazione non solo previene l'overfitting ma guida il modello verso rappresentazioni algoritmiche che generalizzano — un risultato con implicazioni profonde per la comprensione della generalizzazione nel deep learning.

Diffusion Models: Fondamenti Matematici

Il processo di diffusione come SDE stocastica — il forward process, il reverse SDE di Anderson, la score function e l'equivalenza con DDPM.

399.1 · Il Forward Process: Distruggere i Dati con Rumore Crescente

I Diffusion Models (Ho et al. 2020 per DDPM; Song et al. 2021 per la formulazione SDE continua) modellano la distribuzione dei dati come l'endpoint di un processo di inversione di una diffusione stocastica che parte dai dati e li converte progressivamente in rumore gaussiano. Il forward process è una SDE lineare di Ornstein-Uhlenbeck: $d\mathbf{x}_t = -\frac{1}{2}\beta_t \mathbf{x}_t\, dt + \sqrt{\beta_t}\, d\mathbf{W}_t$ dove $\beta_t > 0$ è il noise schedule che controlla la velocità di diffusione. Questa SDE ha soluzione esplicita: la distribuzione marginale di $\mathbf{x}_t$ dato $\mathbf{x}_0$ è gaussiana con media e varianza note: $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)I)$

dove $\bar{\alpha}_t = e^{-\int_0^t \beta_s\, ds}$ decresce da 1 (a $t = 0$, identità) a 0 (a $t = T$, rumore puro). Questa chiusura gaussiana è la chiave della trattabilità del modello: si può campionare $\mathbf{x}_t$ a qualunque tempo t da $\mathbf{x}_0$ in una sola operazione, senza simulare il processo passo dopo passo.

399.2 · Il Reverse Process e il Denoising Score Matching

Il teorema di inversione di Anderson (1982) afferma che il processo inverso di una diffusione stocastica — il processo che va da $\mathbf{x}_T$ (rumore) a $\mathbf{x}_0$ (dati) — è anch'esso una SDE, ma con lo score della distribuzione marginale come drift aggiuntivo:

$$d\mathbf{x}_t = \left[-\frac{1}{2}\beta_t \mathbf{x}_t - \beta_t \nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t)\right] dt + \sqrt{\beta_t}\, d\bar{\mathbf{W}}_t$$

dove $d\bar{\mathbf{W}}_t$ è il moto browniano nel tempo inverso. Questa equazione mostra che per generare campioni dalla distribuzione dei dati è sufficiente conoscere lo score $\nabla_{\mathbf{x}} \log q_t(\mathbf{x})$ per tutti i tempi t — il modello apprendibile. Il denoising score matching (Vincent 2011) fornisce un obiettivo di training stabile: si mostra che lo score della distribuzione marginale $\nabla_{\mathbf{x}} \log q_t(\mathbf{x})$ coincide con l'*aspettativa condizionale del rumore aggiunto*, rescalata: $\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t) = -\boldsymbol{\epsilon}/\sqrt{1-\bar{\alpha}_t}$ dove $\boldsymbol{\epsilon}$ è il rumore gaussiano usato per ottenere $\mathbf{x}_t$ da $\mathbf{x}_0$. Quindi si addestra una rete $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$ a predire il rumore aggiunto: $\mathcal{L}_{\text{DDPM}} = \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\|^2]$. Questa loss è un semplice MSE tra il rumore vero e il rumore predetto — sorprendentemente semplice per un modello con capacità generativa dello stato dell'arte. La generazione campiona $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, I)$ poi inverte il processo applicando iterativamente la denoising step: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t \mathbf{z}$ con $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, I)$.

Inferenza Amortizzata e Variational Autoencoders Avanzati

Oltre il VAE base — VQ-VAE, Hierarchical VAE, β-VAE e la connessione con il disentanglement.

400.1 · Il β-VAE e il Disentanglement

Il VAE standard bilancia la ricostruzione ($\mathbb{E}_q[\log p(x|z)]$) e la regolarizzazione ($\mathrm{KL}(q(z|x)\|p(z))$) con peso unitario. Il β-VAE (Higgins et al. 2017) introduce un coefficiente $\beta > 1$

che aumenta il peso della KL: $\mathcal{L}_{\beta\text{-VAE}} = \mathbb{E}_q[\log p(x|z)] - \beta\,\mathrm{KL}(q(z|x)\|p(z))$. Una KL più grande spinge la distribuzione posteriore verso la prior $p(z) = \mathcal{N}(\mathbf{0}, I)$ — una distribuzione fattorizzata — inducendo disentanglement nelle rappresentazioni latenti: le diverse dimensioni di z catturano fattori di variazione indipendenti e semanticamente interpretabili (forma vs. colore, posizione vs. orientamento). La spiegazione teorica del disentanglement tramite β-VAE è oggetto di dibattito: Chen et al. (2018) mostrano con il "Total Correlation VAE" che la KL si decompone in $\mathrm{KL}(q(z)\|p(z)) + \mathrm{TC}(z) + \sum_i I(z_i; x)$ dove TC è la correlazione totale tra le dimensioni di z, e $\beta > 1$ penalizza la correlazione totale promuovendo la fattorizzazione. Locatello et al. (2019) hanno dimostrato il risultato negativo che senza informazione esplicita sull'identità dei fattori generativi, il disentanglement non è identificabile — qualunque trasformazione invertibile fattorizzata produce la stessa log-likelihood. Il disentanglement è quindi una forma di regolarizzazione implicita, non un obiettivo identificabile senza inductive bias aggiuntivo.

400.2 · VQ-VAE: Spazio Latente Discreto con Quantizzazione Vettoriale

Il VQ-VAE (van den Oord, Vinyals e Kavukcuoglu 2017) sostituisce lo spazio latente continuo gaussiano del VAE con uno spazio latente discreto tramite vector quantization: si mantiene un codebook $\mathbf{E} = \{\mathbf{e}_1, \ldots, \mathbf{e}_K\} \subset \mathbb{R}^d$ di K embedding, e ogni output dell'encoder $\mathbf{z}_e(\mathbf{x})$ viene mappato al codeword più vicino: $\mathbf{z}_q = \mathbf{e}_{k^*}$ dove $k^* = \arg\min_k \|\mathbf{z}_e - \mathbf{e}_k\|$. L'operazione argmin è non differenziabile: per propagare il gradiente si usa il *straight-through estimator* (il gradiente rispetto a $\mathbf{z}_e$ viene passato attraverso la quantizzazione come se fosse un'identità). Il codebook viene aggiornato con l'EMA delle attivazioni degli encoder, garantendo che i codeword si avvicinino ai cluster degli embedding encoder. VQ-VAE produce rappresentazioni discrete compatte che si prestano naturalmente alla modellazione autoregressiva dello spazio latente: un PixelCNN o un Transformer addestrato sul codebook può generare nuove sequenze di codewords, che il decoder converte in output finali. VQ-VAE-2 e la sua estensione gerarchica producono immagini di qualità comparabile a BigGAN (la GAN più potente dell'epoca) con una qualità di compressione notevolmente superiore — il rapporto di compressione tipico è 64× rispetto ai pixel originali, preservando i dettagli semantici rilevanti.

Teoria degli Operatori per il Deep Learning

Reti neurali come operatori tra spazi di funzioni — l'approssimazione di operatori, FNO e la connessione con la geometria differenziale.

401.1 · Il Teorema di Approssimazione Universale per Operatori

Il Teorema di Approssimazione Universale per Operatori (Chen e Chen 1995) afferma che una rete neurale a due layer può approssimare uniformemente qualunque operatore nonlineare continuo $\mathcal{G} : \mathcal{V} \to \mathcal{U}$ tra spazi di Banach, a patto che sia sufficientemente profonda e larga. Formalmente, per qualunque $\varepsilon > 0$ esiste una rete $\mathcal{G}_\theta$ tale che $\sup_{\|v\|_\mathcal{V} \leq 1} \|\mathcal{G}(v) - \mathcal{G}_\theta(v)\|_\mathcal{U} \leq \varepsilon$. Questo teorema generalizza l'approssimazione universale classica (sullo spazio delle funzioni $\mathbb{R}^d \to \mathbb{R}$) agli spazi infinito-dimensionali, giustificando l'uso di reti neurali per apprendere mappature tra funzioni — un obiettivo centrale nei metodi di scienze computazionali basati su NN. Il DeepONet (Lu et al. 2021) implementa questo teorema con un'architettura branch-trunk: la rete branch codifica la funzione di input (campionata in m punti sensor), la rete trunk codifica il punto di query, e l'output è il prodotto interno dei due. Il Fourier Neural Operator (FNO) (Li et al. 2021) parametrizza l'operatore in spazio di Fourier: la convoluzione dell'operatore in spazio fisico diventa una moltiplicazione punto per punto in spazio di Fourier, e si apprende una matrice di mixing dei modi di Fourier $R_\phi \in \mathbb{C}^{d_v \times d_v}$ per i $k_{\max}$ modi di bassa frequenza, producendo un operatore che apprende relazioni non locali in spazio fisico con costo $O(n \log n)$ — lo scaling della FFT. FNO ha dimostrato una generalizzazione eccezionale tra risoluzioni diverse: una rete addestrata sulla soluzione di equazioni di Navier-Stokes a risoluzione 64×64 si applica direttamente a 256×256 senza ri-addestramento, una proprietà chiamata *resolution-invariance* che deriva dalla parametrizzazione in spazio di Fourier indipendente dalla risoluzione della griglia.

Reti Neurali Ricorrenti per Serie Temporali: Apprendimento e Previsione

Modellazione di serie temporali con RNN — la predizione multi-step, il trattamento della non-stazionarietà e la connessione con i modelli di spazio degli stati.

402.1 · La Struttura delle Serie Temporali e le Sfide dell'Apprendimento

Le serie temporali presentano sfide distintive rispetto ai dati i.i.d. del ML supervisionato classico: la dipendenza temporale viola l'assunzione di indipendenza, la non-stazionarietà implica che la distribuzione dei dati cambia nel tempo, e la lungimiranza richiede la propagazione dell'incertezza su orizzonti multipli. Le RNN — e in particolare le LSTM (§382) — affrontano questi problemi mantenendo uno stato interno che condensa la storia rilevante. Per la previsione multi-step, esistono due strategie fondamentalmente diverse. La strategia *direct* addestra un modello separato per ogni orizzonte di previsione h: $\hat{y}_{t+h} = f_h(y_t, y_{t-1}, \ldots)$ con H modelli indipendenti. La strategia *recursive* usa un unico modello a un passo e lo applica iterativamente: $\hat{y}_{t+h} = f(\hat{y}_{t+h-1}, \hat{y}_{t+h-2}, \ldots)$. La strategia diretta evita l'accumulo degli errori di previsione ma perde le dipendenze tra gli orizzonti; la strategia ricorsiva le mantiene ma amplifica gli errori a orizzonti lunghi. La strategia ibrida MIMO (Multi-Input Multi-Output) predice tutti gli H passi in una sola passata, bilanciando i due compromessi.

402.2 · N-BEATS, Temporal Fusion Transformer e PatchTST

N-BEATS (Oreshkin et al. 2020) è un'architettura puramente fully-connected per la previsione di serie temporali che supera i modelli statistici classici (ARIMA, ETS) senza alcuna componente ricorrente o di attenzione: una pila di blocchi residuali che producono decomposizioni interpretabili della serie in componenti di trend e stagionalità. Il Temporal Fusion Transformer (TFT) (Lim et al. 2021) combina attention temporale con variable selection networks per gestire covariate esogene, feature statiche per serie diverse, e feature future note (come le festività), raggiungendo lo stato dell'arte su molti benchmark di previsione retail e energetica. Il PatchTST (Nie et al. 2023) tratta la serie temporale come sequenza di patch (segmenti contigui) e applica un Transformer standard senza positional encoding relativo, raggiungendo performance state-of-the-art con molto meno costo computazionale dei Transformer a livello di singolo passo temporale — la stessa intuizione di ViT (Vision Transformer) applicata alle serie temporali mostra che la localizzazione temporale delle patch migliora l'efficienza dell'attenzione riducendo la lunghezza della sequenza da T a T/p dove p è la dimensione della patch.

Grafi e Graph Neural Networks: Fondamenti Teorici

Messaggi, aggregazione e il teorema di espressività — la connessione con il test di Weisfeiler-Lehman, i GNN spettrali e l'equivarianza sui grafi.

403.1 · Il Framework Message Passing

I Graph Neural Networks (GNN) nel framework di message passing (Gilmer et al. 2017) aggiornano la rappresentazione di ogni nodo aggregando le rappresentazioni dei suoi vicini. Al layer ℓ, il nodo v aggiorna la sua rappresentazione come:

$$\mathbf{h}_v^{(\ell+1)} = \mathrm{UPDATE}^{(\ell)}\Big(\mathbf{h}_v^{(\ell)},\ \mathrm{AGGREGATE}^{(\ell)}\big(\{\mathbf{h}_u^{(\ell)} : u \in \mathcal{N}(v)\}\big)\Big)$$

dove $\mathcal{N}(v)$ è l'insieme dei vicini di v nel grafo. La scelta delle funzioni UPDATE e AGGREGATE determina le proprietà espressive e computazionali del GNN. Le aggregazioni comuni includono: sum (GIN), mean (GCN), max (GraphSAGE), e attention (GAT). Il teorema fondamentale di espressività di Xu et al. (2019) caratterizza il potere discriminativo dei GNN in termini del test di isomorfismo di Weisfeiler-Lehman: qualunque GNN con aggregazione iniettiva (come la somma seguita da una rete neurale universale) è al massimo potente quanto il test WL-1 nel distinguere grafi non isomorfi. Questa limitazione è strutturale — non può essere superata con reti più profonde o più parametri mantenendo la struttura di message passing locale — e ha motivato architetture più potenti come i Higher-Order GNN, i grafi di riga (line graph), e i GNN con distanze geodetiche esplicite.

403.2 · GNN Spettrali: la Convoluzione di Grafo in Spazio di Fourier

I GNN spettrali (Bruna et al. 2014; Kipf e Welling 2017) definiscono la convoluzione di grafo tramite la decomposizione spettrale del Laplaciano normalizzato $\tilde{L} = I - D^{-1/2}AD^{-1/2}$ dove D è la matrice di grado e A la matrice di adiacenza. La GCN di Kipf e Welling usa l'approssimazione del primo ordine del filtro spettrale: $H^{(\ell+1)} = \sigma(\tilde{A}H^{(\ell)}W^{(\ell)})$ dove $\tilde{A} = D^{-1/2}(A+I)D^{-1/2}$ è la matrice di adiacenza normalizzata con self-loops. Questa operazione equivale a una media pesata dei propri embedding e di quelli dei vicini — un passo di *diffusione di feature* sul grafo. Il legame tra la potenza di $\tilde{A}$ (diffusione a k passi) e l'aggregazione a k hop è esatto: $\tilde{A}^k H$ propaga le feature a k hop di distanza, e una GCN a k layer ha un campo

recettivo di k hop — la versione su grafi della receptive field delle CNN su immagini. La differenza fondamentale è che il grafo non ha struttura traslazionalmente invariante, quindi il "filtro" di convoluzione non può essere condiviso spazialmente come nelle CNN: invece, i pesi $W^{(\ell)}$ sono condivisi tra tutti i nodi ma non variano spazialmente all'interno del grafo.

Apprendimento Multi-Task e Trasferimento

Ottimizzazione congiunta di più obiettivi — i conflitti di gradiente, PCGrad, e la teoria del trasferimento come condivisione delle rappresentazioni.

404.1 · La Struttura del Multi-Task Learning e i Conflitti di Gradiente

Nel Multi-Task Learning (MTL) si addestra un unico modello per risolvere T task diversi simultaneamente, condividendo parte delle rappresentazioni tra i task. Il beneficio atteso è il *trasferimento positivo*: task correlati si aiutano a vicenda perché le feature utili per un task sono spesso utili anche per gli altri. Il rischio è il *trasferimento negativo*: task con obiettivi incompatibili si ostacolano a vicenda, portando a performance peggiori rispetto all'addestramento separato. La ragione dei conflitti è ben identificata dalla geometria dei gradienti: se i gradienti di due task $\nabla_\theta \mathcal{L}_1$ e $\nabla_\theta \mathcal{L}_2$ hanno coseno negativo (puntano in direzioni opposte), qualunque aggiornamento dei parametri condivisi che migliora il task 1 peggiora il task 2. Il PCGrad (Yu et al. 2020) risolve i conflitti di gradiente proiettando il gradiente di ogni task sul piano ortogonale al gradiente del task con cui è in conflitto: se $\langle g_1, g_2 \rangle < 0$, si proietta $g_1 \leftarrow g_1 - \frac{\langle g_1, g_2 \rangle}{\|g_2\|^2} g_2$ e analogamente per g_2. L'aggiornamento risultante elimina la componente conflittuale mantenendo le componenti concordanti — un approccio geometricamente pulito che migliora le prestazioni di MTL su molti benchmark di visione e NLP.

404.2 · Trasferimento come Condivisione di Rappresentazioni: la Teoria di Ben-David

La teoria del trasferimento del dominio di Ben-David et al. (2010) fornisce una caratterizzazione quantitativa delle condizioni in cui il trasferimento è benefico. Sia $\mathcal{D}_S$ il dominio sorgente e $\mathcal{D}_T$ il dominio target, e sia h il classificatore addestrato su $\mathcal{D}_S$ e applicato a $\mathcal{D}_T$. Il bound teorico afferma: $\varepsilon_T(h) \leq \varepsilon_S(h) + d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) + \lambda$ dove

ε_T e ε_S sono gli errori sul target e sul sorgente, $d_{\mathcal{H}\Delta\mathcal{H}}$ è la ***distanza di divergenza di dominio*** misurata dalla classe di ipotesi $\mathcal{H}$, e $\lambda = \varepsilon_S(h^*) + \varepsilon_T(h^*)$ è l'errore del classificatore ottimale congiunto. Questo bound identifica tre condizioni per il trasferimento efficace: il modello deve funzionare bene sul sorgente (basso ε_S), il sorgente e il target devono essere vicini nella metrica di divergenza (basso d), e deve esistere un classificatore che funziona bene su entrambi i domini (basso λ). La distanza $d_{\mathcal{H}\Delta\mathcal{H}}$ è la probabilità che un classificatore in $\mathcal{H}$ distingua i campioni dei due domini — precisamente la quantità che gli algoritmi di domain adaptation avversariale (DANN, Ganin e Lempitsky 2015) cercano di minimizzare addestrano un discriminatore a distinguere i due domini e una rete di feature extraction a ingannarlo, riducendo $d_{\mathcal{H}\Delta\mathcal{H}}$ e migliorando il trasferimento.

Interpretabilità e Spiegabilità delle Reti Neurali

Metodi per capire le decisioni dei modelli — SHAP, Grad-CAM, probing classifiers e la distinzione tra interpretabilità locale e globale.

405.1 · SHAP: Valori di Shapley come Attribuzioni Equa

I SHAP values (SHapley Additive exPlanations) (Lundberg e Lee 2017) forniscono attribuzioni delle feature a predizioni individuali con fondamenta nella teoria cooperativa dei giochi. Il valore di Shapley di una feature i per la predizione $\hat{y}$ è la sua contribuzione marginale media al modello, mediata su tutti i possibili ordinamenti delle feature: $\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F|-|S|-1)!}{|F|!}[f(S \cup \{i\}) - f(S)]$ dove F è il set di tutte le feature e $f(S)$ è il valore atteso del modello condizionato solo alle feature in S. I valori di Shapley soddisfano quattro assiomi desiderabili: efficienza ($\sum_i \phi_i = \hat{y} - \mathbb{E}[\hat{y}]$), simmetria (feature intercambiabili hanno lo stesso contributo), linearità (rispetto alla combinazione di modelli), e dummy (feature irrilevanti hanno contributo zero). Il calcolo esatto richiede $2^{|F|}$ valutazioni del modello — esponenziale. TreeSHAP calcola i valori esatti in $O(T \cdot d^2)$ per ensemble di alberi sfruttando la struttura ad albero; KernelSHAP approssima i valori per qualunque modello tramite una regressione pesata locale in $O(k \cdot m)$ dove k è il numero di feature selezionate e m il numero di campioni di background.

405.2 · Grad-CAM e l'Interpretabilità delle CNN

Il Gradient-weighted Class Activation Mapping (Grad-CAM) (Selvaraju et al. 2017) produce mappe di calore che evidenziano le regioni dell'immagine più rilevanti per una predizione di classificazione. Per una classe c, i pesi di importanza del canale k all'ultimo layer convoluzionale sono: $\alpha_k^c = \frac{1}{Z}\sum_{i,j}\frac{\partial y^c}{\partial A_{ij}^k}$ — la media spaziale dei gradienti dell'output rispetto alle attivazioni. La mappa Grad-CAM è la combinazione lineare pesata delle attivazioni con questi pesi, passata attraverso ReLU: $L^c_{\text{GradCAM}} = \text{ReLU}(\sum_k \alpha_k^c A^k)$. La ReLU seleziona solo le attivazioni con contributo positivo alla classe c — le regioni che aumenterebbero la confidenza se amplificate. Grad-CAM++ (Chattopadhyay et al. 2018) estende questo approccio usando derivate seconde per pesare meglio le regioni multiple rilevanti. Il probing classifier è un metodo complementare per l'interpretabilità delle rappresentazioni linguistiche: si addestra un classificatore lineare sui vettori di embedding del modello (congelati) su un task linguistico noto (tagging POS, parsing, coreferenza) — se il classificatore raggiunge alta accuratezza, la rappresentazione "contiene" quella informazione. I risultati dei probing sulle BERT hanno mostrato che diversi layer codificano tipi diversi di informazione linguistica: i layer bassi catturano proprietà morfologiche e sintattiche locali, i layer alti semantica e coreference — una struttura gerarchica emergente analoga agli strati del processamento visivo nelle CNN.

Robustezza Avversariale: Attacchi e Difese

Perturbazioni minime che ingannano i classificatori — FGSM, PGD, certified defenses e il tradeoff robustezza-accuratezza.

406.1 · La Scoperta delle Perturbazioni Avversariali

Szegedy et al. (2014) hanno scoperto che le reti neurali profonde sono vulnerabili a perturbazioni avversariali: modifiche minime e impercettibili all'input (misurate in norma ℓ^∞ o ℓ^2) che cambiano radicalmente la predizione del modello. Una rete che classifica correttamente un'immagine di panda con 99% di confidenza può essere ingannata da una perturbazione di norma $\ell^\infty = 0.01$ (invisibile all'occhio umano) a classificare la stessa immagine come "gibbone" con 99% di confidenza. Il Fast Gradient Sign Method (FGSM) (Goodfellow et al. 2015) è l'attacco più semplice:

$\mathbf{x}_{\text{adv}} = \mathbf{x} + \varepsilon\,\text{sign}(\nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x}, y))$ — si sposta di ε in ogni dimensione nella direzione che massimizza la loss. Nonostante la sua semplicità, FGSM è molto efficace. L'attacco più forte è il PGD (Projected Gradient Descent) (Madry et al. 2018): si eseguono k passi di FGSM con learning rate piccolo, proiettando la perturbazione nella palla ℓ^∞ di raggio ε ad ogni passo — una versione iterativa che trova l'avversario più forte nella palla ammissibile. Madry et al. hanno dimostrato che l'adversarial training con PGD — includere perturbazioni avversariali nel training — è il metodo più efficace di difesa, a spese di un calo di accuratezza sui dati puliti e di un costo computazionale circa k volte superiore al training standard.

406.2 · Certified Defenses e il Tradeoff Robustezza-Accuratezza

Le difese euristiche come l'adversarial training non forniscono garanzie: per qualunque difesa empirica, nuovi attacchi più forti possono essere costruiti. Le certified defenses forniscono garanzie formali: per ogni input $\mathbf{x}$ e ogni perturbazione $\boldsymbol{\delta}$ con $\|\boldsymbol{\delta}\| \leq \varepsilon$, il classificatore produce la stessa classe — impossibile da violare da qualunque attacco entro quel raggio. Il metodo più scalabile è il Randomized Smoothing (Cohen, Rosenfeld e Kolter 2019): si costruisce un classificatore smoothed $g(\mathbf{x}) = \arg\max_c P(f(\mathbf{x} + \boldsymbol{\eta}) = c)$ dove $\boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, \sigma^2 I)$, e si dimostra che g è certificatamente robusto in una palla ℓ^2 di raggio $r = \sigma\Phi^{-1}(p_A - 1/2)$ dove p_A è la probabilità che f produca la classe A sulle versioni rumorose. Il bound fondamentale di Tsipras et al. (2019) dimostra teoricamente che il tradeoff tra accuratezza standard e robustezza avversariale è inevitabile per le distribuzioni naturali: non esiste un modello che sia simultaneamente ottimale per entrambe le metriche quando la distribuzione dei dati ha struttura sfruttabile da un avversario — un risultato con importanti implicazioni per il design di sistemi sicuri.

---

ewpage

Notazione e Convenzioni del Volume VI

Simboli specifici introdotti nel Volume VI, in aggiunta alla notazione generale dell'Appendice A del Volume V.

A.1 · Notazione per i Modelli di Stato e l'Apprendimento per Rinforzo

Nei modelli di spazio degli stati si usano le seguenti convenzioni. Lo stato latente al tempo t è $\mathbf{z}_t \in \mathcal{Z}$; l'osservazione è $\mathbf{x}_t \in \mathcal{X}$; l'azione (nei modelli di controllo) è $\mathbf{a}_t \in \mathcal{A}$. La matrice di transizione del modello lineare è $F \in \mathbb{R}^{d\times d}$; la matrice di osservazione è $H \in \mathbb{R}^{m\times d}$; le covarianze del rumore di processo e di misura sono Q e R. La stima filtrata è $\hat{\mathbf{z}}_{t|t} = \mathbb{E}[\mathbf{z}_t|\mathbf{x}_{1:t}]$; la stima predetta è $\hat{\mathbf{z}}_{t|t-1} = \mathbb{E}[\mathbf{z}_t|\mathbf{x}_{1:t-1}]$; la matrice di covarianza dell'errore di stima è $P_{t|t}$ e $P_{t|t-1}$ rispettivamente. Il guadagno di Kalman è $K_t \in \mathbb{R}^{d\times m}$; l'innovazione è $\boldsymbol{\nu}_t = \mathbf{x}_t - H\hat{\mathbf{z}}_{t|t-1}$.

Nell'apprendimento per rinforzo si usano le seguenti convenzioni. Un MDP è una tupla $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$ con spazio degli stati $\mathcal{S}$, spazio delle azioni $\mathcal{A}$, funzione di transizione $P : \mathcal{S}\times\mathcal{A} \to \Delta(\mathcal{S})$ (distribuzione sugli stati successori), funzione di ricompensa $R : \mathcal{S}\times\mathcal{A} \to \mathbb{R}$, e fattore di sconto $\gamma \in [0, 1)$. La politica stocastica è $\pi : \mathcal{S} \to \Delta(\mathcal{A})$; la politica deterministica è $\mu : \mathcal{S} \to \mathcal{A}$. La funzione valore di stato è $V^\pi(s) = \mathbb{E}_\pi[G_t|S_t = s]$ con $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$; la funzione Q è $Q^\pi(s, a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a]$. La funzione di vantaggio è $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$. L'errore di differenza temporale a un passo è $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$; il ritorno di n passi è $G_t^{(n)} = \sum_{k=0}^{n-1} \gamma^k R_{t+k+1} + \gamma^n V(S_{t+n})$.

A.2 · Notazione per l'Online Learning e i Banditi

Nel framework OCO (Online Convex Optimization), il giocatore sceglie $\mathbf{x}_t \in \mathcal{K}$ e subisce una perdita $f_t(\mathbf{x}_t)$ dove $f_t : \mathcal{K} \to \mathbb{R}$ è convessa. Il regret cumulativo dopo T round è $\mathrm{Regret}_T = \sum_{t=1}^{T} f_t(\mathbf{x}_t) - \min_{\mathbf{x}\in\mathcal{K}} \sum_{t=1}^{T} f_t(\mathbf{x})$. Nel problema dei banditi a K braccia, $\mu_k = \mathbb{E}[r_k]$ è il reward medio dell'azione k; $\mu^* = \max_k \mu_k$ è il reward ottimale; $\Delta_k = \mu^* - \mu_k$ è il gap subottimale; $n_k(t)$ è il numero di volte che l'azione k è stata scelta fino al round t; $\hat{\mu}_k(t)$ è la stima empirica del reward di k al round t. Il regret cumulativo è $R(T) = \mu^* T - \sum_{t=1}^{T} \mu_{I_t}$ dove I_t è l'azione scelta al round t.

A.3 · Notazione per la Teoria PAC e la Generalizzazione

Un problema di apprendimento PAC è specificato dalla classe di ipotesi $\mathcal{H}$, dallo spazio degli input $\mathcal{X}$, dallo spazio degli output $\mathcal{Y}$, dalla distribuzione sconosciuta $\mathcal{D}$ su $\mathcal{X}\times\mathcal{Y}$, dai parametri di accuratezza $\varepsilon > 0$ e confidenza $\delta \in (0, 1)$. La perdita vera (rischio) è $L_{\mathcal{D}}(h) = \mathbb{P}_{(x,y)\sim\mathcal{D}}[h(x) \neq y]$; la perdita empirica su m campioni è $\hat{L}_S(h) = \frac{1}{m}\sum_{i=1}^{m} \mathbf{1}[h(x_i) \neq y_i]$. La VC dimension di $\mathcal{H}$ è indicata con $d_{\mathrm{VC}}(\mathcal{H})$ o semplicemente d_{VC}. La Rademacher Complexity empirica di $\mathcal{H}$ su S è $\hat{\mathcal{R}}_S(\mathcal{H}) =$

$\mathbb{E}_\sigma[\sup_{h\in\mathcal{H}} \frac{1}{m}\sum_i \sigma_i h(x_i)]$ **con** $\sigma_i \in \{-1,+1\}$ **i.i.d. uniformi. Nel framework PAC-Bayes,** P **è la distribuzione prior fissata prima dei dati e** Q **è la distribuzione posterior appresa.**

A.4 · Identità Utili per i Processi Stocastici

L'equazione di Fokker-Planck per la densità $p_t(\mathbf{z})$ **di una SDE** $d\mathbf{z} = \boldsymbol{\mu}(\mathbf{z},t)dt + \sigma(\mathbf{z},t)d\mathbf{W}$ **è:** $\frac{\partial p_t}{\partial t} = -\nabla\cdot(\boldsymbol{\mu}p_t) + \frac{1}{2}\nabla^2(\sigma^2 p_t)$**. Per la SGLD con prior** $p_0(\theta) \propto e^{-U_0(\theta)}$ **e likelihood** $\prod_i p(x_i|\theta)$**, la distribuzione stazionaria è la posterior** $\pi(\theta) \propto e^{-U(\theta)}$ **con** $U(\theta) = U_0(\theta) - \sum_i \log p(x_i|\theta)$**. Il mixing time di una catena di Markov reversibile su spazio discreto con spazio degli stati** $\mathcal{S}$**, misura stazionaria** π**, e spettral gap** γ **soddisfa:** $t_{\text{mix}}(\varepsilon) \leq \frac{1}{\gamma}\log\frac{1}{\varepsilon\pi_{\min}}$ **dove** $\pi_{\min} = \min_{x\in\mathcal{S}}\pi(x)$**. Per le catene di Markov a tempo continuo, il tasso di convergenza alla stazionarietà sotto la disuguaglianza di Poincaré con costante** C_P **è** $\|p_t - \pi\|_{\text{TV}}^2 \leq e^{-2t/C_P}\|p_0 - \pi\|_{\text{TV}}^2$**.**

Algoritmi in Pseudocodice

Riepilogo degli algoritmi principali introdotti nel Volume VI.

Algoritmo: Algoritmo 1 — Filtro di Kalman

*// **Input:** **F**, **H**, **Q**, **R**,* $\pi_0, \Sigma_0; osservazioni x_1, \ldots, xT$

Inizializza: $_{1|0} = \pi_0, P_{1|0} = \Sigma_0$

per t = 1,. . . , T:

*// **Fase di correzione***

$\boldsymbol{K}_t = P_{t|t-1}H^T(HP_{t|t-1}H^T + R)^{-1}$

$_{t|t} = \hat{z}_{t|t-1} + K_t(x_t - H\hat{z}_{t|t-1})$

$\boldsymbol{P}_{t|t} = (I - K_tH)P_{t|t-1}$

*// **Fase di predizione***

$_{t+1|t} = F\hat{z}_{t|t}$

$\boldsymbol{P}_{t+1|t} = FP_{t|t}F^T + Q$

***Algoritmo: Algoritmo 2 — Q-Learning con** $\varepsilon - greedy$*

*// **Input: MDP ignoto, learning rate** $\alpha, sconto\gamma, esplorazione\varepsilon$*

***Inizializza: Q(s,a) = 0** $\forall s, a; sceglis_1$*

per t = 1,...,T:

***con prob.** $\varepsilon : a_t \leftarrow azionecasuale$; **altrimenti** $: a_t \leftarrow argmax_a Q(s_t, a)$*

***Esegui** $\boldsymbol{a}_t; osservar_{t+1,s_{t+1}}$*

$\boldsymbol{Q(s}_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma max_{a'} Q(s_{t+1,a'}) - Q(s_t, a_t)]$

$\boldsymbol{s}_t \leftarrow s_{t+1}$

Algoritmo: Algoritmo 3 — PPO (Proximal Policy Optimization)

*// **Input: policy** $\pi_\theta, criticV\phi$, **clip** $\varepsilon \approx 0.2, GAE\lambda$*

per iterazione = 1,...:

***Raccogli T passi di esperienza con** $\pi_{\theta_o}ld$*

***Calcola vantaggi** $\hat{\boldsymbol{A}}_t conGAE(\gamma, \lambda)$*

per K epoche di ottimizzazione:

$\boldsymbol{r}_t(\theta) = \pi_\theta(a_t|s_t)/\pi_{\theta_old(a_t|s_t)}$

$\boldsymbol{L\hat{}CLIP} = \boldsymbol{E}_t[min(r_{tt}, clip(r_t, 1-\varepsilon, 1+\varepsilon)_t)]$

$\boldsymbol{L\hat{}VF} = \boldsymbol{E}_t[(V\phi(\boldsymbol{s}_t) - V_{\phi}\Theta arg)\Theta 2]$

Aggiorna θ,ϕ ***massimizzando L^CLIP - c***$_1 \Theta VF + c_2 H[\pi_\theta]$

$\theta_old \leftarrow \theta$

Algoritmo: Algoritmo 4 — Soft Actor-Critic (SAC)

// Due critici $Q_{\theta_1}, Q_{\theta_2}$***; policy*** π_ϕ***; temperatura*** α ***(adattiva)***

Inizializza: replay buffer D, reti target $\boldsymbol{Q}_{\theta 1, Q_{\theta 2}}$

per ogni passo:

$\boldsymbol{a}_t\ \pi_\phi(\cdot|s_t); osservar_t, s_{t+1;aggiungiaD}$

Campiona batch (s,a,r,s') da D

$\boldsymbol{y} = \boldsymbol{r} + \gamma(min_i Q_{\theta i(s',')-\alpha log\pi_\phi('|s')),'\ \pi_\phi}$

Aggiorna $\theta_i : minE[(Q_{\theta i(s,a)-y})\Theta 2]$

Aggiorna $\phi : maxE[min_i Q_{\theta i(s,)-\alpha log\pi_\phi(|s)]}$

Aggiorna $\alpha : minE[-\alpha(log\pi_\phi(|s) + H)]$

Aggiorna reti target con EMA: $\theta \leftarrow \tau\theta + (1-\tau)\theta$

Algoritmo: Algoritmo 5 — Split Conformal Prediction

// Input: modello f, calibration set $(\boldsymbol{x}_i, y_i)_{i=1}\Theta m, livello\alpha$

Calcola nonconformity scores: $\boldsymbol{s}_i = |y_i - f(x_i)|peri = 1, \ldots, m$

Calcola soglia: q = quantile(1-$\alpha)(m+1)/mdeglis_i$

per ogni nuovo $x_{m+1:}$

$\hat{C}(x_{m+1}) = [f(x_{m+1}) - q, f(x_{m+1}) + q]$

Garanzia: $P(y_{m+1} \in \hat{C}(x_{m+1})) \geq 1 - \alpha, perqualunquedistribuzionescambiabile$

Algoritmo: Algoritmo 6 — CMA-ES (semplificato)

// **Input: f da minimizzare, dimensione d,** $\lambda campioni, \mu selezionati$

Inizializza: $m \in, \sigma = 0.3, C = I, p_\sigma = 0, p_c = 0$

ripeti fino a convergenza:

per k=1,..., $\lambda : x_k \ N(m, \sigma^2 C); valuta f(x_k)$

Ordina per f; seleziona i $\mu migliori : x_{1:\mu}$

$m \leftarrow \Sigma_{i=1}^{\mu} w_i x_i$

Aggiorna $p_\sigma, C(evoluzionecumulativa), \sigma(adattamentopasso)$

ritorna m

Algoritmo: Algoritmo 7 — Thompson Sampling per Banditi Bernoulli

// **Prior Beta(**$\alpha_k, \beta_k) = Beta(1, 1) \forall k$

per t = 1,...,T:

per k = 1,...,K: campiona $\theta_k \ Beta(\alpha_k, \beta_k)$

$I_t \leftarrow argmax_k \theta_k$

Osserva $r_t \in 0, 1$

se $r_t = 1 : \alpha_{I_t} \leftarrow \alpha_{I_t} + 1$

altrimenti: $\beta_{I_t} \leftarrow \beta_{I_t} + 1$

Tabelle di Complessità e Confronto tra Algoritmi

Riepilogo comparativo dei costi computazionali dei principali metodi del Volume VI.

C.1 · Modelli di Stato e Filtri

Metodo	*Costo per passo*	*Memoria*	*Esatto?*	*Note*
Filtro di Kalman	$O(d^3)$	$O(d^2)$	***Sì (lin.-gauss.)***	***Inversione matrice*** $m \times$
EKF	$O(d^3)$	$O(d^2)$	***No (approx.)***	***Calcolo jacobiano***
UKF	$O(d^3)$	$O(d^2)$	***No (3° ordine)***	$2d+1$ ***sigma-points***
Filtro Particellare	$O(Nd)$	$O(Nd)$	***Asintotico in*** N	N ***= numero particelle***
HMM Forward	$O(K^2)$	$O(K)$	***Sì***	K ***stati latenti***
Viterbi	$O(K^2)$	$O(KT)$	***Sì (MAP)***	***Con backpointer***
Baum-Welch (EM)	$O(K^2T)$***/iter***	$O(KT)$	***No (locale)***	***Per sequenza di lunghe***

C.2 · Algoritmi di Apprendimento per Rinforzo

Algoritmo	*Tipo*	*Spazio stati*	*Costo/passo*	*Garanzia teorica*
Q-Learning tabellare	***Off-policy***	***Discreto finito***	$O(1)$	***Convergenza a*** Q^*
DQN	***Off-policy***	***Continuo/grande***	$O(Wd)$***/update***	***Nessuna (appross***
REINFORCE	***On-policy PG***	***Qualunque***	$O(T)$***/episodio***	***Convergenza loca***
A3C	***On-policy AC***	***Qualunque***	$O(n_{\text{workers}}T)$	***Convergenza asin***
PPO	***On-policy***	***Qualunque***	$O(K_{\text{epoch}}T)$	***Monotonia appro***
SAC	***Off-policy***	***Continuo***	$O(W)$***/update***	***Ottimo max-entro***
RMAX	***Model-based***	***Discreto finito***	$O(S^2A)$	***PAC-MDP*** ε***-otti***

C.3 · Bound di Regret nell'Online Learning

Algoritmo	*Setting*	*Regret*	*Lower bound*
OGD	***OCO convesso***	$O(\sqrt{T})$	$\Omega(\sqrt{T})$
OGD fortemente convesso	***OCO s.c.***	$O(\log T)$	$\Omega(\log T)$
Hedge/EWA	***Esperti***	$O(\sqrt{T \log N})$	$\Omega(\sqrt{T \log N})$
Fixed Share	***Tracking,*** S ***cambi***	$O(\sqrt{T(S \log N + S \log T/S)})$	$\Omega(\sqrt{TS \log N})$
UCB1	***Bandit stocastico***	$O(\sqrt{KT \log T})$	$\Omega(\sqrt{KT})$
EXP3	***Bandit avversariale***	$O(\sqrt{KT \log K})$	$\Omega(\sqrt{KT})$
Thompson Sampling	***Bandit stocastico***	$O(\sum_k \log T/\Delta_k)$	$\Omega(\sum_k \log T/\Delta_k$
LinUCB	***Bandit contestuale***	$O(d\sqrt{T \log T})$	$\Omega(d\sqrt{T})$
Universal Portfolio	***Portfolio online***	$O(\log T)$ ***(log-wealth)***	$\Omega(\log T)$
Hedge ritardato (d)	***Esperti + ritardo***	$O(\sqrt{T(d+1) \log N})$	$\Omega(\sqrt{Td \log N})$

Teoremi di Impossibilità e Risultati Negativi Fondamentali

I teoremi che delimitano i confini di ciò che è apprendibile, equo, identificabile e computazionalmente trattabile.

D.1 · Impossibilità nell'Equità Algoritmica

Il teorema di impossibilità di Chouldechova (2017) afferma che per qualunque classificatore binario su due gruppi con tassi di base diversi $P(Y = 1|A = 0) \neq P(Y = 1|A = 1)$***, le tre condizioni seguenti non possono essere soddisfatte simultaneamente: (1) calibrazione*** $P(Y = 1|\hat{p}, A = a) = \hat{p}$ ***per*** $a \in \{0, 1\}$***; (2) bilanciamento dei falsi positivi*** $P(\hat{Y} = 1|Y = 0, A = 0) = P(\hat{Y} = 1|Y = 0, A = 1)$***; (3) bilanciamento dei falsi negativi*** $P(\hat{Y} = 0|Y = 1, A = 0) = P(\hat{Y} = 0|Y = 1, A = 1)$***. Il teorema di Kleinberg, Mullainathan e Raghavan (2016) formula la stessa impossibilità in termini di risk score calibrati. La conseguenza pratica è che qualsiasi sistema di predizione per decisioni ad alto impatto (concessione di prestiti, libertà vigilata, screening medico) deve scegliere quale nozione di equità privilegiare — una scelta di valore che va esplicitata, non nascosta dietro la tecnicità del modello.***

D.2 · Impossibilità nell'Identificabilità Causale

Il teorema di non-identificabilità lineare gaussiana afferma che per modelli causali lineari con tutti i rumori gaussiani, la distribuzione osservazionale è compatibile con entrambe le direzioni causali $X \to Y$ e $Y \to X$ — non è possibile stabilire la direzione causale senza interventi o assunzioni aggiuntive. Questo non vale per i modelli non gaussiani (LiNGAM, §373) o non lineari con rumore additivo (ANM). Il teorema di non-parametric identifiability (Peters, Mooij et al. 2014) quantifica precisamente le eccezioni: il set di accoppiamenti (P_X, f) per cui la direzione causale non è identificabile ha misura di Lebesgue zero nello spazio delle distribuzioni e delle funzioni — la non-identificabilità è genericamente impossibile fuori dal caso lineare gaussiano.

D.3 · Impossibilità nella Sample Complexity

Il teorema No Free Lunch (Wolpert 1992, versione apprendimento) afferma che per qualunque algoritmo di apprendimento A esiste una distribuzione $\mathcal{D}$ su cui A non fa meglio dell'indovinare a caso, per qualunque numero finito di esempi. Più precisamente, per qualunque A e m finito, $\max_{\mathcal{D}} L_{\mathcal{D}}(A(S_m)) \geq 1/2 - 1/|m\mathcal{X}|$. La conseguenza è che la generalizzazione richiede assunzioni implicite o esplicite sulla struttura del problema — non esiste un algoritmo universalmente buono senza inductive bias. Il lower bound di Lai-Robbins (1985) afferma che qualunque algoritmo bandit consistente deve avere $\liminf_{T\to\infty} R(T)/\log T \geq \sum_{k:\Delta_k>0} \Delta_k / D(\mu_k \| \mu^*)$ dove D è la divergenza KL tra le distribuzioni di reward — nessun algoritmo può fare meglio del regret logaritmico in modo asintoticamente ottimale.

---

ewpage

Riferimenti Bibliografici del Volume VI

Opere fondative, articoli originali e monografie per le sezioni §342–§395.

Sezione A — Modelli di Stato e Filtraggio

Baum, L.E. & Petrie, T. (1966). Statistical inference for probabilistic functions of finite state Markov chains. Annals of Mathematical Statistics, 37(6), 1554–1563. — Il paper fondativo degli HMM.

Rabiner, L.R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE, 77(2), 257–286. — Il tutorial di riferimento per le applicazioni degli HMM.

Kálmán, R.E. (1960). A new approach to linear filtering and prediction problems. Journal of Basic Engineering, 82(1), 35–45. — Il paper originale del filtro di Kalman.

Rauch, H.E., Tung, F. & Striebel, C.T. (1965). Maximum likelihood estimates of linear dynamic systems. AIAA Journal, 3(8), 1445–1450. — Il Kalman Smoother (algoritmo RTS).

Julier, S.J. & Uhlmann, J.K. (1997). A new extension of the Kalman filter to nonlinear systems. SPIE Proceedings on Signal Processing, Sensor Fusion, and Target Recognition, 182–193. — Il Filtro Unscented.

Gordon, N.J., Salmond, D.J. & Smith, A.F.M. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. IEE Proceedings F, Radar and Signal Processing, 140(2), 107–113. — Il bootstrap particle filter originale.

Hafner, D. et al. (2019). Dream to control: Learning behaviors by latent imagination. ICLR 2020. — Dreamer con RSSM.

Hafner, D. et al. (2023). Mastering diverse domains through world models. arXiv:2301.04104. — DreamerV3.

Sezione B — Apprendimento per Rinforzo

Bellman, R. (1957). Dynamic Programming. Princeton University Press. — La programmazione dinamica e l'equazione di Bellman.

Sutton, R.S. (1988). Learning to predict by the methods of temporal differences. Machine Learning, 3(1), 9–44. — Il TD learning.

Watkins, C.J.C.H. & Dayan, P. (1992). Q-learning. Machine Learning, 8(3–4), 279–292. — La prova di convergenza del Q-learning.

Mnih, V. et al. (2015). Human-level control through deep reinforcement learning. Nature, 518, 529–533. — DQN.

Williams, R.J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning, 8(3–4), 229–256. — REIN-

FORCE.

Schulman, J., Moritz, P., Levine, S., Jordan, M. & Abbeel, P. (2016). High-dimensional continuous control using generalized advantage estimation. ICLR 2016. — GAE.

Mnih, V. et al. (2016). Asynchronous methods for deep reinforcement learning. ICML 2016. — A3C.

Schulman, J., Levine, S., Moritz, P., Jordan, M. & Abbeel, P. (2015). Trust region policy optimization. ICML 2015. — TRPO.

Schulman, J. et al. (2017). Proximal policy optimization algorithms. arXiv:1707.06347. — PPO.

Haarnoja, T., Zhou, A., Abbeel, P. & Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning. ICML 2018. — SAC.

Levine, S. (2018). Reinforcement learning and control as probabilistic inference: Tutorial and review. arXiv:1805.00909. — Control as Inference.

Brafman, R.I. & Tennenholtz, M. (2002). R-max — A general polynomial time algorithm for near-optimal reinforcement learning. JMLR, 3, 213–231. — RMAX e PAC-MDP.

Jaksch, T., Ortner, R. & Auer, P. (2010). Near-optimal regret bounds for reinforcement learning. JMLR, 11, 1563–1600. — UCRL2.

Sezione C — Teoria PAC e Generalizzazione

Valiant, L.G. (1984). A theory of the learnable. Communications of the ACM, 27(11), 1134–1142. — Il framework PAC originale.

Vapnik, V.N. & Chervonenkis, A.Y. (1971). On the uniform convergence of relative frequencies of events to their probabilities. Theory of Probability and Its Applications, 16(2), 264–280. — Il teorema VC.

Bartlett, P.L., Harvey, N., Liaw, C. & Mehrabian, A. (2019). Nearly-tight VC-dimension and Pseudodimension bounds for piecewise linear neural networks. JMLR, 20(63), 1–17. — VC dimension delle reti ReLU.

Koltchinskii, V. & Panchenko, D. (2002). Empirical margin distributions and

bounding the generalization error of combined classifiers. Annals of Statistics, 30(1), 1–50. — Rademacher Complexity per classifier combinati.

McAllester, D.A. (1998). Some PAC-Bayesian theorems. COLT 1998, 230–234.

McAllester, D.A. (1999). PAC-Bayesian model averaging. COLT 1999, 164–170.

Dziugaite, G.K. & Roy, D.M. (2017). Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many parameters. UAI 2017. — PAC-Bayes per reti neurali reali.

Littlestone, N. & Warmuth, M.K. (1986). Relating data compression and learnability. Technical Report, UCSC. — Compression bounds.

Zhang, C., Bengio, S., Hardt, M., Recht, B. & Vinyals, O. (2017). Understanding deep learning requires rethinking generalization. ICLR 2017. — Memorizzazione con etichette casuali.

Belkin, M., Hsu, D., Ma, S. & Mandal, S. (2019). Reconciling modern machine-learning practice and the classical bias–variance trade-off. PNAS, 116(32), 15849–15854. — La doppia discesa.

Papyan, V., Han, X.Y. & Donoho, D.L. (2020). Prevalence of neural collapse during the terminal phase of deep learning training. PNAS, 117(40), 24652–24663. — Neural Collapse.

Sezione D — Mixture of Experts

Jacobs, R.A., Jordan, M.I., Nowlan, S.J. & Hinton, G.E. (1991). Adaptive mixtures of local experts. Neural Computation, 3(1), 79–87. — Il MoE originale.

Shazeer, N. et al. (2017). Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. ICLR 2017. — Sparse MoE con top-k e load balancing.

Fedus, W., Zoph, B. & Dean, J. (2022). Switch transformer: Scaling to trillion parameter models with simple and efficient sparsity. JMLR, 23(120), 1–39. — Switch Transformer.

Mixtral team, Mistral AI. (2023). Mixtral of experts. arXiv:2401.04088. — Mixtral $8\times7B$.

Zhou, Y. et al. (2022). Mixture-of-experts with expert choice routing. NeurIPS

2022. — Expert Choice routing.

Puigcerver, J. et al. (2023). From sparse to soft mixtures of experts. arXiv:2308.00951. — Soft MoE.

Riquelme, C. et al. (2021). Scaling vision with sparse mixture of experts. NeurIPS 2021. — V-MoE.

Sezione E — Kernel Methods Avanzati

Rahimi, A. & Recht, B. (2007). Random features for large-scale kernel machines. NeurIPS 2007. — Random Fourier Features.

Williams, C.K.I. & Seeger, M. (2001). Using the Nyström method to speed up kernel machines. NeurIPS 2001. — Approssimazione di Nyström.

Wilson, A.G., Hu, Z., Salakhutdinov, R. & Xing, E.P. (2016). Deep kernel learning. AISTATS 2016. — DKL.

Garnelo, M. et al. (2018). Neural processes. ICML Workshop on Theoretical Foundations and Applications of Deep Generative Models. — Neural Processes.

Shervashidze, N., Schweitzer, P., Van Leeuwen, E.J., Mehlhorn, K. & Borgwardt, K.M. (2011). Weisfeiler-Lehman graph kernels. JMLR, 12, 2539–2561.

Johnson, W. & Lindenstrauss, J. (1984). Extensions of Lipschitz mappings into a Hilbert space. Contemporary Mathematics, 26, 189–206. — Il lemma JL.

Sezione F — Algoritmi Evolutivi

Hansen, N. & Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation, 9(2), 159–195. — CMA-ES.

Salimans, T., Ho, J., Chen, X., Sidor, S. & Sutskever, I. (2017). Evolution strategies as a scalable alternative to reinforcement learning. arXiv:1703.03864. — OpenAI-ES.

Wierstra, D. et al. (2014). Natural evolution strategies. JMLR, 15, 949–980. — NES.

Stanley, K.O. & Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. Evolutionary Computation, 10(2), 99–127. — NEAT.

Real, E. et al. (2019). Regularized evolution for image classifier architecture search. AAAI 2019. — AmoebaNet.

Such, F.P. et al. (2017). Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. arXiv:1712.06567.

Sezione G — Causal Discovery Moderno

Zheng, X., Aragam, B., Ravikumar, P. & Xing, E.P. (2018). DAGs with NO TEARS: Continuous optimization for structure learning. NeurIPS 2018. — NOTEARS.

Yu, Y., Chen, J., Gao, T. & Yu, M. (2019). DAG-GNN: DAG structure learning with graph neural networks. ICML 2019.

Shimizu, S., Hoyer, P.O., Hyvärinen, A. & Kerminen, A. (2006). A linear non-Gaussian acyclic model for causal discovery. JMLR, 7, 2003–2030. — LiNGAM.

Schölkopf, B. et al. (2021). Toward causal representation learning. Proceedings of the IEEE, 109(5), 612–634.

Arjovsky, M., Bottou, L., Gulrajani, I. & Lopez-Paz, D. (2019). Invariant risk minimization. arXiv:1907.02893. — IRM.

Hauser, A. & Bühlmann, P. (2012). Characterization and greedy learning of interventional Markov equivalence classes of directed acyclic graphs. JMLR, 13, 2409–2464. — GIES.

Chernozhukov, V. et al. (2018). Double/debiased machine learning for treatment and structural parameters. The Econometrics Journal, 21(1), C1–C68. — Double ML.

Callaway, B. & Sant'Anna, P.H.C. (2021). Difference-in-differences with multiple time periods. Journal of Econometrics, 225(2), 200–230. — DiD eterogeneo.

Sezione H — Teoria della Misura e Apprendimento Teorico

Kolmogorov, A.N. (1933). Grundbegriffe der Wahrscheinlichkeitsrechnung. Springer. — La fondazione assiomatica della probabilità.

Doob, J.L. (1953). Stochastic Processes. Wiley. — Il teorema di convergenza per martingali.

Billingsley, P. (1999). Convergence of Probability Measures. 2nd edition, Wiley. — Il TCL funzionale (Teorema di Donsker).

Jacot, A., Gabriel, F. & Hongler, C. (2018). Neural tangent kernel: Convergence and generalization in neural networks. NeurIPS 2018. — NTK.

Mei, S., Montanari, A. & Nguyen, P.-M. (2018). A mean field view of the landscape of two-layer neural networks. PNAS, 115(33), E7665–E7671. — Mean field theory per reti neurali.

Vovk, V., Gammerman, A. & Shafer, G. (2005). Algorithmic Learning in a Random World. Springer. — La monografia fondativa della Conformal Prediction.

Angelopoulos, A.N., Bates, S., Candès, E.J., Jordan, M.I. & Lei, L. (2022). Learn then test: Calibrating predictive algorithms to achieve risk control. arXiv:2110.01052. — Conformal Risk Control.

Hardt, M., Price, E. & Srebro, N. (2016). Equality of opportunity in supervised learning. NeurIPS 2016. — Equalized odds.

Chouldechova, A. (2017). Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. Big Data, 5(2), 153–163. — Teorema di impossibilità dell'equità simultanea.

Candès, E.J., Romberg, J. & Tao, T. (2006). Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. IEEE Transactions on Information Theory, 52(2), 489–509. — Compressed Sensing.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. JRSS-B, 58(1), 267–288. — LASSO.

Cohen, T.S. & Welling, M. (2016). Group equivariant convolutional networks. ICML 2016. — Il teorema di caratterizzazione delle convoluzioni equivarianti.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. & Smola, A. (2017). Deep sets. NeurIPS 2017. — Il teorema di universalità per funzioni di insiemi.

Oord, A.V.D., Li, Y. & Vinyals, O. (2018). Representation learning with contrastive predictive coding. arXiv:1807.03748. — InfoNCE e CPC.

Chen, T., Kornblith, S., Norouzi, M. & Hinton, G. (2020). A simple framework

for contrastive learning of visual representations. ICML 2020. — SimCLR.

Radford, A. et al. (2021). Learning transferable visual models from natural language supervision. ICML 2021. — CLIP.

Lawrence, N.D. (2004). Gaussian process latent variable models for visualisation of high dimensional data. NeurIPS 2004. — GPLVM.

Kearns, M. & Valiant, L. (1994). Cryptographic limitations on learning Boolean formulae and finite automata. Journal of the ACM, 41(1), 67–95. — Hardness of learning e crittografia.